
MI COMPUTER
27
CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR
Editorial Delta, S.A.
150 ptas.

MI COMPUTER CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona, y comercializado en exclusiva por Distribuidora Olimpia, S.A., Barcelona

Volumen III - Fascículo 27

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones, Francisco Martín
Asesor técnico: Jesús Nebra

Redactores y colaboradores: G. Jefferson, R. Ford, S. Tarditti, A. Cuevas, F. Blasco

Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, Barcelona-8
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador,* se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-85822-94-3 (tomo 3)
84-85822-82-X (obra completa)
Depósito Legal: B. 52/1984

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 188407
Impreso en España - Printed in Spain - Julio 1984

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, Madrid-34.
Distribuye para Argentina: Viscontea Distribuidora, S.C.A., La Rioja 1134/56, Buenos Aires.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio Blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Ferrenquín a Cruz de Candelaria, 178, Caracas, y todas sus sucursales en el interior del país.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 16 690 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 087 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 3371872 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Distribuidora Olimpia (Paseo de Gracia, 88, 5.°, Barcelona-8), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Distribuidora Olimpia, en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

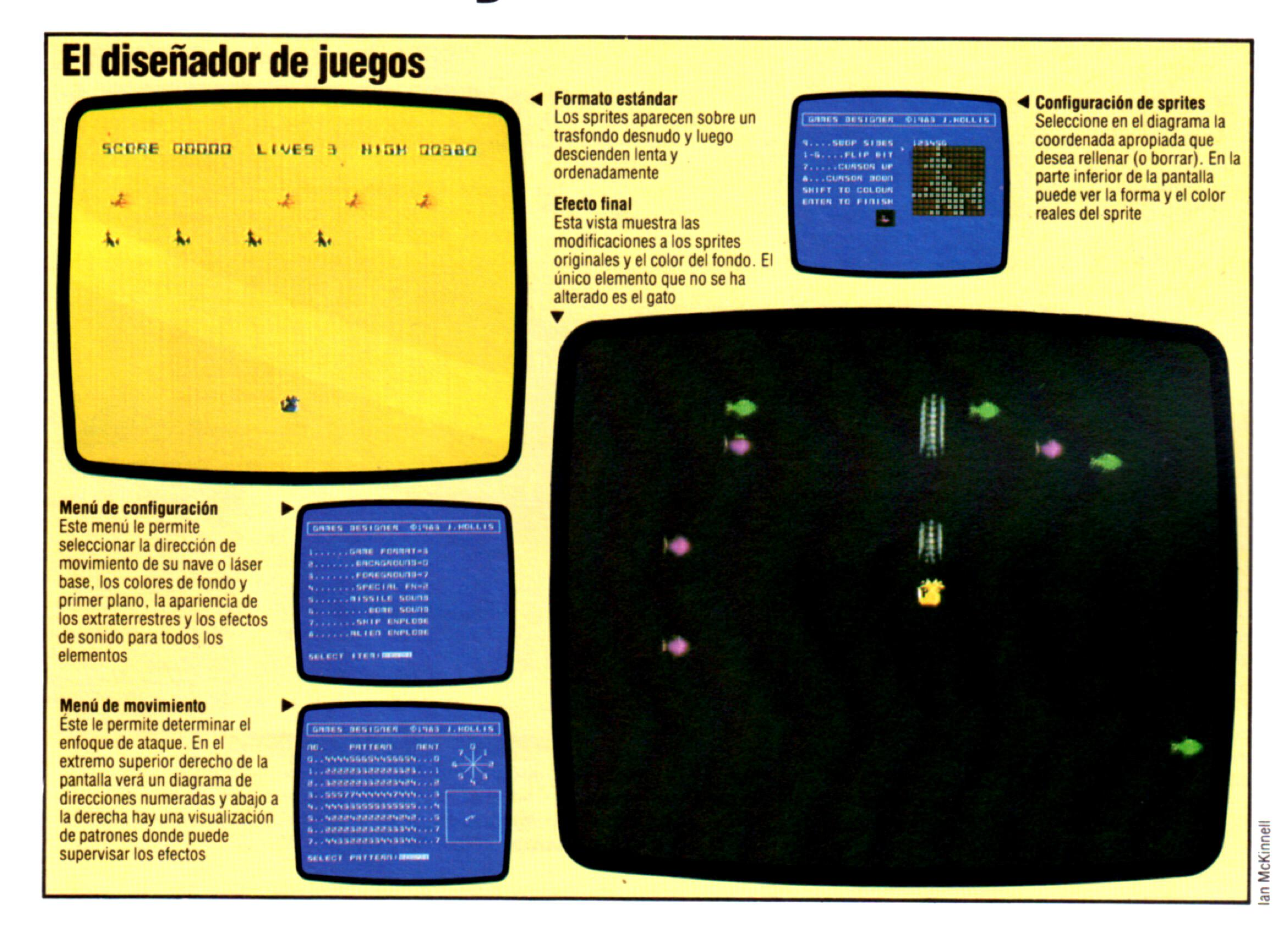
No se efectúan envíos contra reembolso.

Con este fascículo se incluyen las portadillas correspondientes al segundo volumen (fascículos 13-24)



Aventura y fantasía

El diseñador de juegos



Formato estándar
Los sprites aparecen sobre un trasfondo desnudo y luego descienden lenta y ordenadamente

Efecto final
Esta vista muestra las modificaciones a los sprites originales y el color del fondo. El único elemento que no se ha alterado es el gato

Configuración de sprites
Seleccione en el diagrama la coordenada apropiada que desea rellenar (o borrar). En la parte inferior de la pantalla puede ver la forma y el color reales del sprite

Menú de configuración
Este menú le permite seleccionar la dirección de movimiento de su nave o láser base, los colores de fondo y primer plano, la apariencia de los extraterrestres y los efectos de sonido para todos los elementos

Menú de movimiento
Éste le permite determinar el enfoque de ataque. En el extremo superior derecho de la pantalla verá un diagrama de direcciones numeradas y abajo a la derecha hay una visualización de patrones donde puede supervisar los efectos

Ian McKinnell

Existen paquetes para juegos que permiten al usuario formular las reglas del juego que ha diseñado y dar vuelo a su imaginación

La mayoría de los juegos para ordenadores personales entran en dos categorías: juegos de aventuras con participación, con o sin una representación gráfica del escenario en la pantalla, como *El Hobbit*, y los "simuladores de fantasía" totalmente basados en pantalla, como *Space invaders* o *Asteroids*.

Incluso un análisis superficial de los dos tipos genéricos revela la causa de su similitud. Tomando en primer lugar los simuladores de fantasía, el juego estilo *Space invaders* exige dos requisitos previos: una base de disparo y un blanco. De modo que si pudiéramos construir versiones abstractas de estas dos cosas y permitir que el diseñador de juegos decida de forma independiente acerca de la posición del defensor, la potencia de fuego y la frecuencia e intensidad de las ráfagas de ataque, entonces sería factible producir diversos juegos, cada uno de los cuales se diferenciaría en alguna sutileza respecto al siguiente, simplemente variando los parámetros. Un análisis de los numerosos juegos, cuyos nombres suenan parecidos, producidos en una "generación", revela que los productores profesionales de software han aplicado este criterio.

Un ejemplo excelente de un paquete de juegos que le permite al usuario hacer exactamente eso es *Games designer* (Diseñador de juegos), de John Hollis (de Software Studios/Quicksilva, para el Spectrum de 48 K). El *Games designer* ofrece ocho juegos esbozados. Después de seleccionar uno de ellos, usted puede alterar todos los parámetros básicos citados, pero no construir un juego nuevo. El paquete es activado por menú.

Después de escoger un juego, que será el bosquejo sobre el cual se efectuarán las modificaciones, se le ofrecen unas opciones del tipo y la envergadura de dicha modificación, empezando por la

forma y el color de los sprites utilizados para representar a los protagonistas. Aun cuando se presenta un menú de los sprites existentes, esto no es una limitación sino más bien una comodidad, porque el usuario puede comenzar por un diseño predefinido y modificar cualquier bit dentro de la matriz de 12 × 12, u optar por empezar con una "página en blanco" y originar una figura completamente nueva. El formato de 12 × 12 (dos caracteres por dos caracteres) da como resultado un sprite perfectamente utilizable y característico.

La definición de sprites es activada por menú, visualizándose las opciones de las órdenes disponibles a la izquierda de la pantalla, y una imagen ampliada del sprite en cuestión a la derecha. Un agregado muy útil es una segunda imagen del sprite, que se visualiza al pie del texto, en el color y las dimensiones reales.

El siguiente paso consiste en definir de qué modo se desplazarán estos sprites alrededor del campo para juegos. Esta sección, denominada *configuración*, le permite al jugador mayores posibilidades que simplemente dirigir el movimiento de distintivos individuales. Desde este momento, por el orden en el cual se visualizan las opciones del menú principal, se lo invita a seleccionar un patrón de movimiento para los distintivos de pantalla. Para hacer más complicado el movimiento, en esta etapa se pueden unir entre sí dos o más patrones.

Ahora el usuario debe decidir la frecuencia de las ondas de ataque e insertar efectos especiales, tanto visuales como sonoros.

Habiendo creado un juego, sólo falta guardarlo en cassette para que se pueda volver a jugar a él en el futuro. Y aquí el paquete fracasa estrepitosamente, desde el punto de vista del usuario, ya que no almacena un juego en forma representable, sino sólo como parámetros de entrada para el paquete *Games designer* propiamente dicho, haciendo que el ejercicio resulte completamente inútil para quienes desearían desarrollar un juego para después venderlo.

Una pluma creativa

En contraposición directa con el *Games designer, The quill* (La pluma de ave) está dirigido a quienes desean crear juegos de aventuras e intentar comercializarlos; en el manual aparece incluso una sección titulada *Para vender su aventura*, que proporciona valiosas indicaciones acerca de los pasos que habría que dar para la verificación y depuración. Una de las principales sugerencias consiste en hacer que la mayor cantidad posible de personas pruebe el juego con sentido crítico. Hay que destacar que los autores sólo exigen que "se mencione en algún lugar [del juego] que se lo escribió con *The quill*", una rara muestra de altruismo.

Aunque se lo presenta como un generador de juegos del tipo *Dungeons and dragons* (Calabozos y dragones), *The quill* utiliza técnicas más parecidas a las que emplean los paquetes comerciales para administración de bases de datos. Se divide en tres partes principales: la base de datos propiamente dicha; un editor de base de datos, que permite establecer los parámetros, y un intérprete de base de datos, que ejecuta el juego de forma interactiva.

Así como uno podría catalogar *Space invaders* y todos los demás juegos similares como juegos protagonistas-posicionamiento, del mismo modo la base de los juegos de aventuras es el "laberinto conceptual": un laberinto que existe en otras dimensiones aparte de las puramente espaciales. De este modo, además de buscar el camino que conduce a la salida, al jugador se le presenta una lista de objetos, cada uno de los cuales sólo es útil en una circunstancia específica; si usted nunca entra en una habitación donde hay un portalámparas vacío, por ejemplo, no tiene ningún sentido que lleve consigo una bombilla.

Para dotar de mayor interés al juego, se suele limitar el número de objetos que el jugador puede llevar consigo de un lugar a otro, pero no hay impedimentos para que el diseñador utilice este límite artificial para hacer ir y venir al jugador acumulando los objetos que necesitará para realizar las tareas que se le asignan a medida que avanza el juego.

Definiendo el objeto
The quill ofrece a los usuarios la posibilidad de crear juegos de aventuras. De estilo similar a algunos paquetes para administración de bases de datos, exige que se definan las posiciones y los objetos que se incluyen en la aventura. Después, las técnicas de programación orientadas hacia un objeto permiten llamar a estos atributos asignados y visualizarlos, o utilizarlos como parámetros cada vez que el objeto o posición aparezca en el juego



Ian McKinnell

La primera etapa en el diseño de un juego de aventuras consiste en crear el guión y el escenario en que se desarrollará. El manual de *The quill* comienza esbozando un juego sencillo que enfrenta al jugador con una opción de diez objetos en un entorno simple que abarca seis compartimientos. Este juego no está incorporado en el programa, lo que obliga al usuario a dar entrada a todos los parámetros bosquejados en el manual antes de poder jugar a él, tal como tendría que hacer si usted hubiera creado su propio juego.

The quill pone un límite al número de parámetros que se pueden utilizar en el juego, pero éste es tan elevado (252 posiciones y 210 objetos) que es muy poco probable que se quede sin opciones.

El paquete puede producir juegos de aventuras que compitan con la mayoría de los juegos disponibles a nivel comercial (de hecho, cierta cantidad de ellos se han escrito a partir de él), pero tiene sus limitaciones. En primer lugar, no permite la creación de gráficos en pantalla como los que tan bien se emplean en *El hobbit*, por ejemplo; tampoco permite ninguna clase de interacción con personajes, de modo que no tiene ningún sentido intentar crearlos. Además de ofrecerle al usuario de juegos de aventuras un método simplificado para componer sus propios juegos, su mayor mérito reside en la disciplina que requiere su utilización.

Nacido para triunfar

Desde su aparición, en 1982, se ha apoderado del mercado de microprocesadores de 16 y 32 bits

El 68000 guarda un fuerte parecido con el MC6800, un microprocesador Motorola algo más viejo que todavía se sigue utilizando mucho, en especial para periféricos tales como controladores inteligentes. Esto significa que el 68000, mucho más capaz, es fácil de conectar en interface y tiene el respaldo de una amplia variedad de hardware prefabricado. Éste incluye tableros de E/S con chips PIA (*Peripheral Interface Adaptor*: adaptador para interface de periféricos) 6821, unidades de representación visual (VDU) con CRTC (*Cathode Ray Tube Controllers*: controladores de tubo de rayos catódicos) 6845, relojes que utilizan los sincronizadores programables 6840 y controladores de disco.

Otra configuración del 68000 hace que les resulte atractivo a los diseñadores de ordenadores: la anchura de sus buses de datos y direcciones. Éstos están completamente separados el uno del otro, y cada bit posee su propia patilla, a diferencia del 8086, 8088 y Z8000, en los cuales las patillas están multiplexadas entre sí: los dos buses comparten un juego de patillas y las señales se intercalan y se decodifican en su punto de destino.

El procesador, por consiguiente, puede operar tan rápidamente como se lo permita el resto del sistema, y con los chips de RAM más recientes de 50 o 90 nanosegundos (10^{-9} segundos), esto significa una reducción, o incluso la desaparición, de los estados de espera. El procesador más veloz de la serie 68000 es el 68000L12, muchos de cuyos componentes se pueden hacer funcionar a 14 Megahertzios.

Sinclair Research ha utilizado el sucesor del 68000, el 68008, para el QL. Internamente es muy similar a los otros de su misma serie, pero, para hacerlo más compatible con los sistemas de ocho bits existentes, posee un bus de datos de ocho bits en lugar del bus de 16 bits de anchura total. Dado que necesita menos patillas, viene en un paquete normal en número de 40.

Pronto la Motorola va a producir un microprocesador aún más potente. El 68020 es un microprocesador de 32 bits que requiere un paquete de 96 patillas, cuya forma y estilo aún están por definir. El 68881, un procesador especializado en matemática de coma flotante con ocho registros (cada uno de 80 bits de ancho) y que aumentará notablemente la cantidad de datos "reales" que se puedan manipular, está también en fase de planificación.

Otros chips de la serie 68000 proporcionan funciones de E/S similares a las que ofrecían chips más antiguos, aunque muy mejoradas. Pero desde el punto de vista del programador, el 68000 posee muchas ventajas en relación a la mayoría de los otros procesadores de 16 bits, debido a la simetría de sus registros de direcciones y de datos y al rico juego de instrucciones.

Aunque tampoco es perfecto. En primer lugar, se hace una distinción entre los registros de direcciones y los de datos, a pesar de que son del mismo tamaño (32 bits) y, en la mayoría de los casos, se opera sobre ellos de la misma manera mediante las mismas instrucciones. A consecuencia de ello, a menudo es necesario desplazar datos desde un registro de direcciones a un registro de datos, manipularlos y luego devolverlos al registro de direcciones. Habría sido más sencillo si Motorola hubiera hecho posible utilizar cualquier registro tanto para datos como para direcciones.

En segundo lugar, hay redundancias en el juego de instrucciones, pero como éstas son producto de lo que podría denominarse "cruce de modalidad de direccionamiento", no tienen mayores consecuencias. En realidad, este fenómeno se produce porque las diversas modalidades de direccionamiento son tan distintas que en algunas ocasiones una puede significar exactamente lo mismo que otra, a pesar de haber llegado a ella mediante instrucciones diferentes.

No obstante, en general la serie Motorola 68000 proporciona unas CPU amplias, veloces y eficaces, que se están utilizando cada vez más. En 1983 fueron empleadas en el Lisa y el Mackintosh de Apple, en el QL de Sinclair y en muchas máquinas de gestión para usuarios múltiples de menor proyección en el mercado. Al proporcionar configuraciones que hace sólo un par de años hubieran costado un buen número de billetes, y al estar disponibles a un precio razonable, parecen estar llamadas a hacerse populares entre la nueva generación de máquinas, tal como lo son en la actualidad el Z80 y el 6502.



6502
Desarrollado por MOS Technology, el microprocesador 6502 habría de convertirse, con el Z80 de Zilog, en el soporte de la industria de microordenadores. Utiliza un bus de direcciones de 16 bits y un bus de datos de 8 bits. De todas sus peculiaridades, la que más destaca es la organización de sus registros. Hay un solo acumulador, pero se puede utilizar toda la página de memoria 0 como registros para usos generales

68000
Motorola volvió a diseñar y a desarrollar el 6800 para crear el 6809, pero no llegó a tiempo para asegurar para el mismo un amplio sector del mercado de 8 bits. Esto habría de resultar ventajoso, dado que indujo a la empresa a desarrollar el procesador 68000, de 16/32 bits. El 68000 puede utilizar muchos de los chips de apoyo de las series 6502/6800 y está construido alrededor de ocho registros de datos de 32 bits y siete registros de direcciones de 32 bits

Z80
Teóricamente más potente que el MOS 6502, el Zilog Z80 utiliza estructuras de buses de datos y direcciones similares, pero posee un juego de registros considerablemente más fuertes (12 registros de 8 bits para fines generales y dos registros índice de 16 bits) y un juego de instrucciones mucho más amplio. Quizá su mayor ventaja respecto al 6502 sea su capacidad para apoyar el sistema operativo CP/M

Kevin Jones

Reparaciones

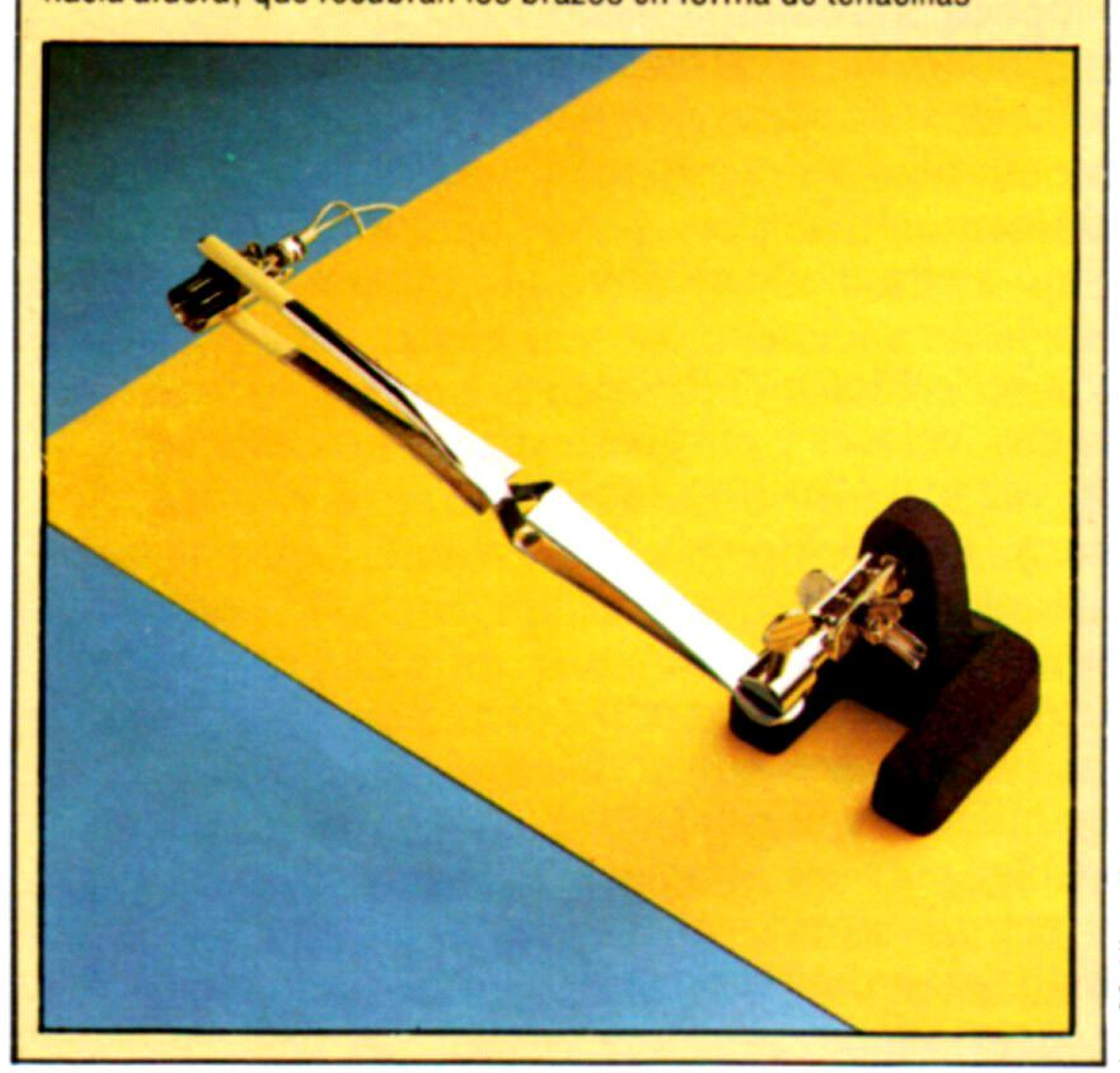
Comenzamos una serie de artículos dedicados a las reparaciones y añadidos mediante soldadura. El principiante podrá así efectuar él mismo esos trabajos que tan caros suelen cobrar los profesionales

Tanto la soldadura corriente como la fuerte son métodos para unir entre sí dos objetos de metal mediante una aleación metálica blanda (que, por tanto, se funde con facilidad); para los trabajos de electrónica se emplean el plomo y el estaño. Los objetos a unir se calientan a una temperatura superior al punto de fusión de la soldadura (280 °C), aplicamos el soldador a los componentes, los unimos y retiramos la fuente de calor. A medida que se enfrían, la soldadura se solidifica.

Un hierro para soldar aplicado directamente a una barra de soldadura la fundirá casi de inmediato. La soldadura caliente se enfriará casi en el acto al entrar en contacto con el componente frío. El resultado se conoce como juntura "seca". En el mejor de los casos, no permanecerá adherido. En el peor, establecerá una conexión muy pobre, que tal vez permita incluso un flujo intermitente de electricidad. Sólo existe una forma de evitar esta juntura imperfecta: calentar el componente hasta que el soldador se funda al contacto.

Eficiencia artesana

Una de las razones por las cuales los ordenadores forman parte de nuestra vida cotidiana es su reducido tamaño. Los elementos que los componen, por tanto, son tan diminutos que trabajar con ellos puede ser problemático. Existen varias prensas de tornillo y pinzas ligeras a un precio razonable. Si una pinza no agarra lo suficiente, hágale unos manguitos de cinta adhesiva con la goma hacia afuera, que recubran los brazos en forma de tenacillas



Ian McKinnell

Soldador
Es importante recordar que muchos componentes electrónicos son sensibles al sobrecalentamiento. La respuesta al problema que ello supone reside en la elección del soldador, el tamaño de su punta y el diámetro de la soldadura utilizada. Para fines generales, como juntar conductores y conectores, esta elección es menos crucial; un soldador de 15 o 20 vatios y una soldadura de núcleos múltiples de alrededor de 1,5 mm son adecuados para la mayoría de las tareas

Pelar cables
Siempre que se utilizan cables, primero es necesario quitar el aislamiento y la cobertura con destreza y pulcritud. Los limpiadores de cables sencillos, como el que vemos aquí, son muy baratos y se pueden prefijar para una determinada profundidad de corte, quitando limpiamente el aislamiento pero dejando intacto el cable



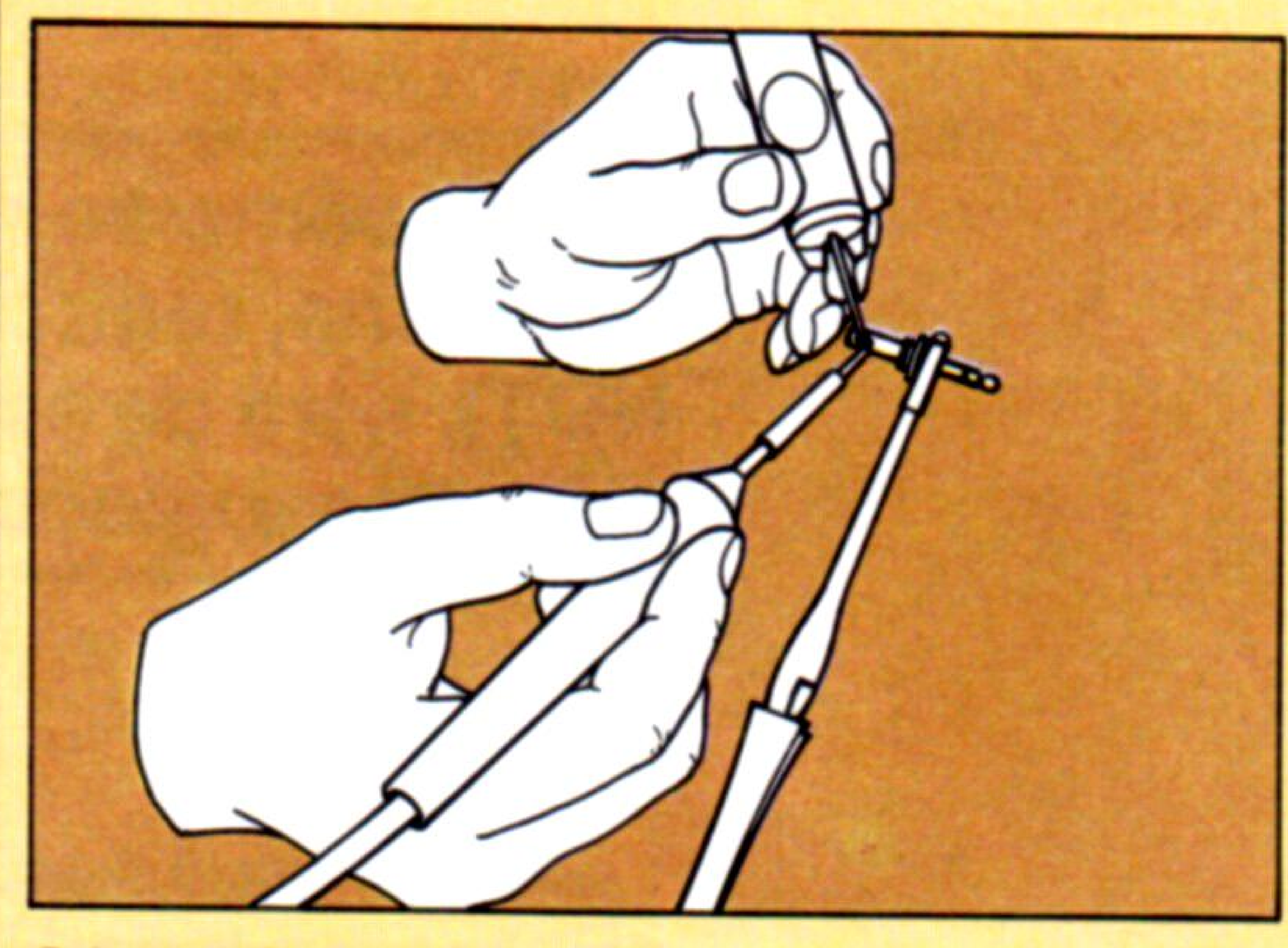
El destornillador adecuado
Existen dos variedades de destornillador: el corriente y el cruciforme, aunque en este último caso es necesario diferenciar entre el Philips y el Pozidriv. Por los tamaños en que se opera con la mayoría de los ordenadores personales, los dos modelos son intercambiables. Es probable que usted llegue a necesitar tanto el corriente como el cruciforme, y quizá en varios tamaños

Apretar fuerte
Para el principiante bastan dos clases de alicates: los alicates combinados, que pueden también utilizarse para cortar alambre en los casos más duros, y los redondos, mucho más ligeros, para las tareas que requieran un trato más delicado

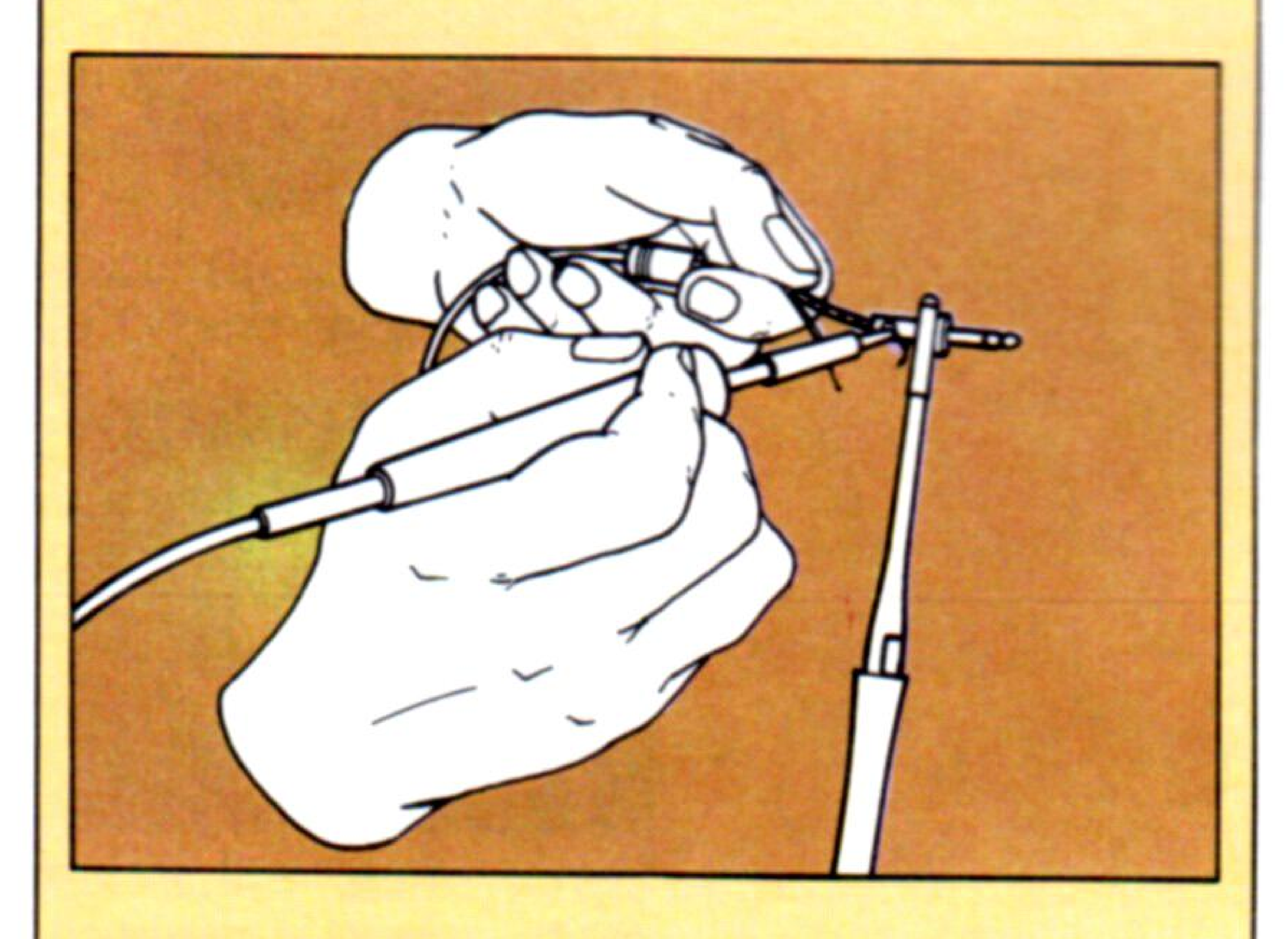
Punto de corte
Las herramientas para cortar alambres son de dos clases principales, que se conocen como fresas de disco, como el par que vemos en la ilustración, y cortadores en punta, cuyas puntas de corte están en 90° respecto al plano de los mangos. En una emergencia, un simple cortauñas podría servir para la tarea; ¡pero no espere que después de usarlo pueda volver a cortarse las uñas!

Michael Brownlow

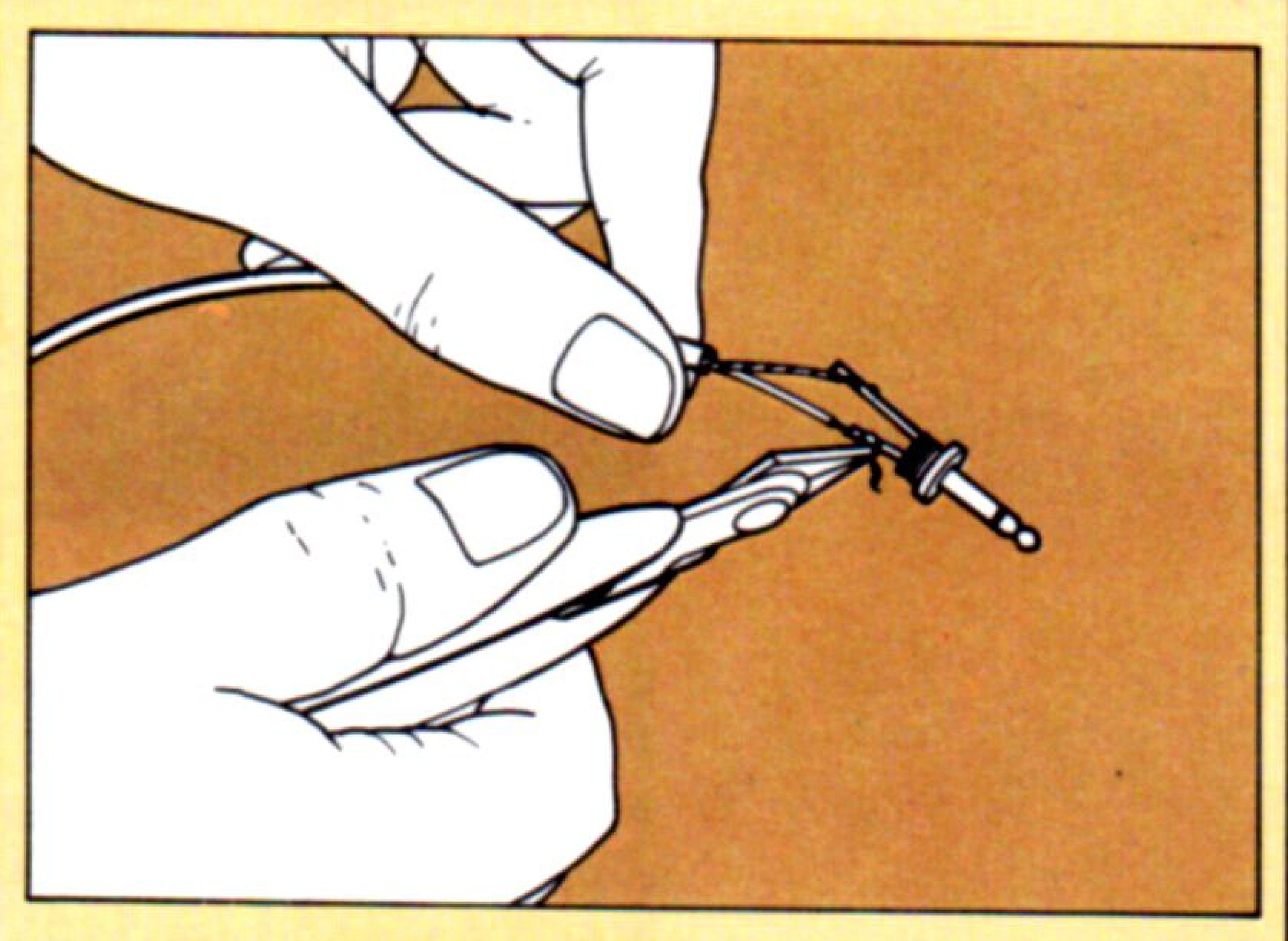
Soldar un enchufe



Pele y estañe
La primera etapa del proceso para formar una conexión consiste en pelar los alambres del cable retirando el aislamiento. No se quede corto. Pele más alambre del que necesite y recórtelo después. Sostenga el cable en la prensa del tornillo y caliéntelo con el soldador. Aplique la soldadura al cable y cuando lo comience a pasar utilice el soldador para guiarlo limpiamente a través de toda la superficie expuesta del cable. Este proceso, que se conoce como estañado, simplifica el posterior proceso de soldar; para entonces la soldadura estará sobre el componente. Repita esto mismo con el enchufe. Cuando el terminal todavía esté lo suficientemente caliente como para hacer correr la soldadura, aplique el cable estañado, quite la fuente de calor cuando la soldadura tanto del componente como del cable se haya fundido y esté unida, sóplelas para enfriarlas por debajo del punto de fusión, y la juntura ya está hecha



Manteniendo todo en orden
La siguiente etapa de la tarea consiste en recortar el material de desecho. Recorte los cables y también las patillas terminales de los componentes; pero sólo en el último momento, es decir, después de haber verificado la tarea. La razón del recorte es sencilla: un extremo largo de cable inservible, o una patilla terminal que sobresalga podrían hacer contacto con alguna otra cosa, lo que produciría un cortocircuito o, peor todavía, uno de esos enervantes fallos intermitentes

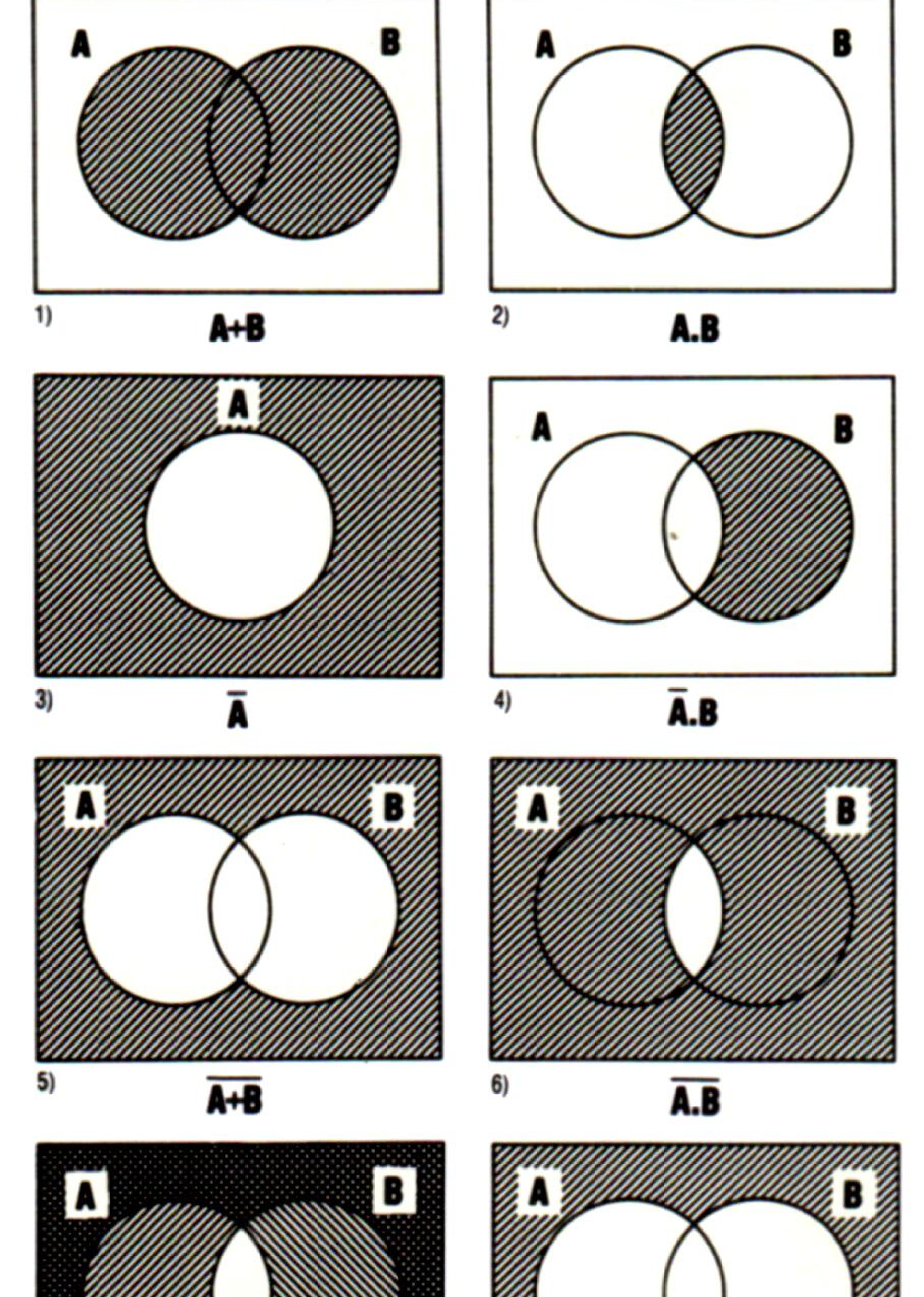


Kevin Jones

Refinando el proceso

Saber cómo se simplifican expresiones complejas de álgebra booleana, reduciéndolas a un mínimo de operadores (AND, OR, NOT), será de especial importancia cuando se apliquen a los circuitos lógicos

Los diagramas de Venn son un complemento gráfico valioso para la simplificación de expresiones de álgebra booleana, al permitir dibujar el resultado de una expresión como zonas sombreadas. La superficie dentro de un rectángulo (con símbolo 1 = identidad o conjunto universal) representa todas las combinaciones posibles de valores verdaderos de las entradas, y los círculos dentro del rectángulo corresponden a combinaciones determinadas. A continuación ofrecemos algunas representadas en diagramas de Venn:



1) A+B — 2) A.B — 3) $\overline{A}$ — 4) $\overline{A}.B$ — 5) $\overline{A+B}$ — 6) $\overline{A.B}$ — 7) $\overline{A}+\overline{B}$ — 8) $\overline{A}.\overline{B}$

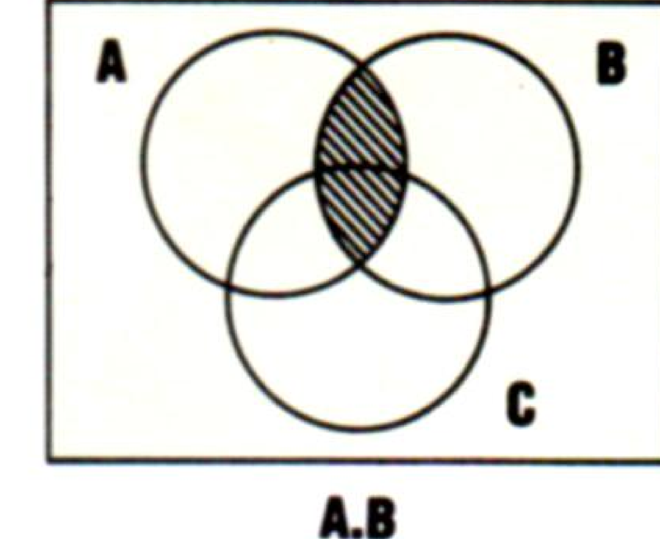
Comparando los diagramas 5 y 7 salta a la vista que NOT(A OR B) no es lo mismo que NOT(A) OR NOT(B). Del mismo modo, los diagramas 6 y 8 demuestran que NOT(A AND B) no equivale a NOT(A) AND NOT(B).

Tal vez la forma más sencilla de imaginar AND y OR en términos de diagramas de Venn sea pensando que A.B es la superficie común a A y a B; y que A + B es la unión de superficies de A y B. Varias relaciones son evidentes en el álgebra booleana. Para cada una de ellas usted podría intentar construir un diagrama de Venn como muestra de que son verdaderas. (0 representa el *conjunto vacío*, es decir, una combinación imposible.) He aquí estas seis relaciones evidentes:

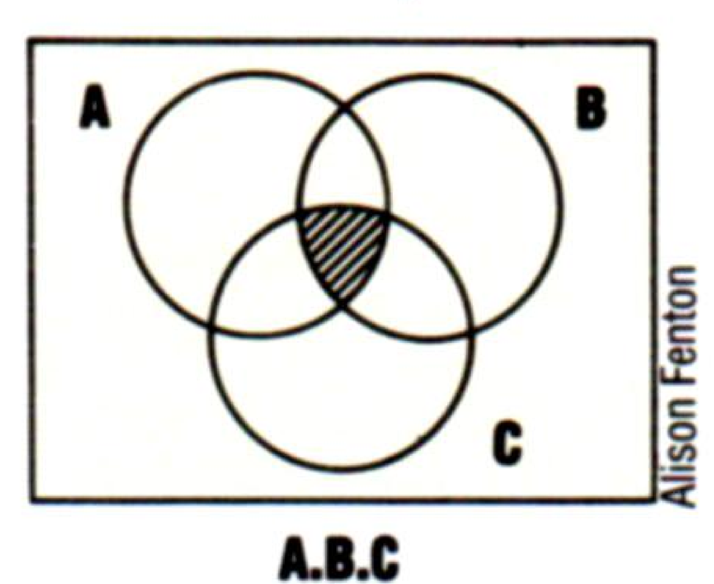
1) $A.A = A$
2) $A.\overline{A} = 0$
3) $A.0 = 0$
4) $A.1 = A$
5) $A.(A + B) = A$
6) $A.(\overline{A} + B) = A.B$

Leyes del álgebra booleana

El concepto de *dualidad* es otra valiosa, y esta vez intrigante, ayuda para la simplificación, que se basa en la simetría de los operadores AND y OR. Para formar la dual verdadera de cualquier relación verdadera, cambie todos los AND por OR y viceversa y, del mismo modo, todos los ceros por unos y viceversa. Por ejemplo, tomemos la quinta relación de las incluidas en la lista precedente. La dual de esta relación sería A + A.B = A. Esta expresión también es verdadera. Demuestra el importante principio de la *absorción*. Mirando un diagrama de Venn, es fácil ver que el término A.B cae enteramente dentro de A y, por consiguiente, se puede decir que ha sido absorbido por A. Esta idea se puede ampliar para un caso de tres variables, como A.B + A.B.C. = A.B. El par de diagramas de Venn que ofrecemos ilustra la última expresión:



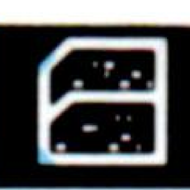
A.B



A.B.C

Alison Fenton

Por otra parte, vuelva a observar los diagramas de Venn del principio. Comparando los diagramas 5 y 8, vemos que la siguiente relación importante siempre es verdadera: $\overline{A + B} = \overline{A}.\overline{B}$. Comparando los diagramas 6 y 7 vemos que: $\overline{A.B} = \overline{A} + \overline{B}$. Estas



dos relaciones son las célebres *leyes de Morgan* y se pueden aplicar a casos más complicados como éste de tres variables: $(\overline{A + B + C} = \bar{A}.\bar{B}.\bar{C}$ y $\overline{A.B.C.} = \bar{A} + \bar{B} + \bar{C})$. Las leyes de Morgan se pueden aplicar por etapas:

$(\overline{A + B}).\bar{C}$

$= \bar{A}.\bar{B}.\bar{C}$ (según Morgan, diagramas 5 y 7, aplicado al paréntesis)

$= \overline{A + B + C}$ (según Morgan, diagramas 6 y 8, tomando como un solo componente el paréntesis)

Existen además tres propiedades del álgebra normal que se pueden aplicar al álgebra booleana. La *propiedad asociativa* permite trastrocar los paréntesis:

$(A.B).C = A.(B.C) = A.B.C.$

$(A + B) + C = A + (B + C) = A + B + C$

El orden en que se escriben las letras se puede alterar por la *propiedad conmutativa*:

$A.B. = B.A.$

$A + B = B + A$

La *propiedad distributiva* permite multiplicar los paréntesis:

$A.(B + C) = A.B + A.C$

Ejemplos de simplificación

1) Simplificar $(\overline{A + B} + \bar{A}.B).B$

$= (\bar{A}.\bar{B} + \bar{A}.B).B$ (Morgan)

$= \bar{A}.\bar{B}.B + \bar{A}.B.B.$ (propiedad distributiva)

$= 0 + \bar{A}.B$ ($\bar{B}.B = 0$, $B.B = B$)

$= \bar{A}.B$

2) Simplificar $\bar{A}.\bar{B} + \bar{A}.B + A.B$

$= \bar{A}.(\bar{B} + B) + A.B$ (propiedad distributiva)

$= \bar{A} + A.B$ ($\bar{B} + B = 1$)

$= \bar{A} + B$ (dual de relación 6)

3) Simplificar $\overline{\bar{A} + B} + \overline{\bar{A} + \bar{B}} + \bar{A}.B$

$= \bar{\bar{A}}.\bar{B} + \bar{\bar{A}}.\bar{\bar{B}} + \bar{A}.B$ (Morgan)

$= A.\bar{B} + A.B + \bar{A}.B$ ($\bar{\bar{A}} = A$)

$= A.(\bar{B} + B) + \bar{A}.B$ (propiedad distributiva)

$= A + \bar{A}.B$ ($\bar{B} + B = 1$)

$= A + B$ (dual de relación 6)

Una puerta XOR simplificada

En el capítulo anterior analizamos un circuito no simplificado para una puerta del OR-exclusivo (o sea, XOR). Volvamos ahora a examinar el mismo problema, pero esta vez sabemos cómo simplificar la expresión booleana resultante y, por tanto, el circuito. La tabla de verdad para la puerta XOR era:

ENTRADA		SALIDA
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

Dijimos que esta tabla de verdad previamente se resumía así: $C = \bar{A}.B + A.\bar{B}$. Aquí poca simplificación se puede hacer, y debemos resignarnos a usar un circuito de cinco puertas. Pero hay un enfoque alternativo para abordar este problema. A partir de la tabla de verdad se puede decir que C es 1 si A y B no son ambos 1 o ambos 0. En términos booleanos podemos escribir:

$C = \overline{A.B + \bar{A}.\bar{B}}$

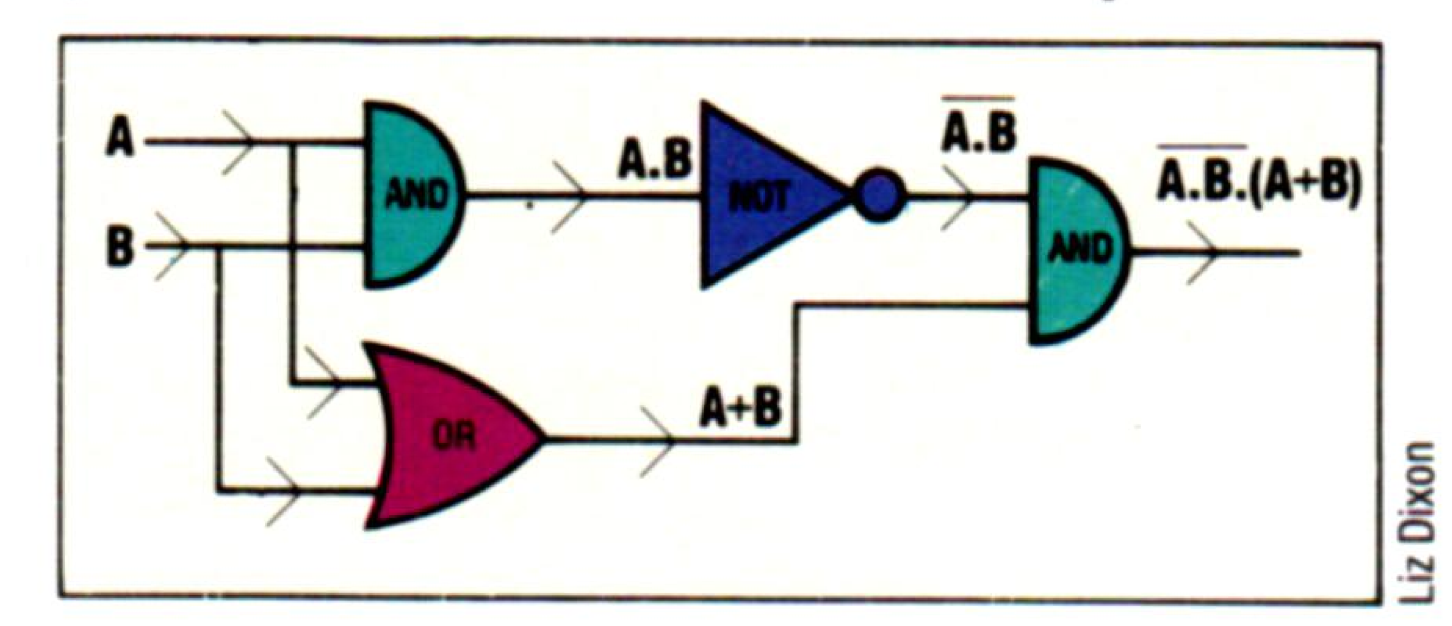
Utilizando repetidamente las leyes de Morgan, podemos simplificar el circuito para obtener:

$C = (\overline{A.B}).(\overline{\bar{A}.\bar{B}})$

y por último:

$C = \overline{A.B}.(A + B)$

que sólo necesita un circuito de cuatro puertas:



Liz Dixon

Circuito sumador completo

Anteriormente analizamos el proceso de la suma binaria y diseñamos un circuito simple para sumar dos bits que tuviera dos salidas, una para el dígito de la suma parcial y otra para llevo. A este circuito lo denominamos sumador medio. Si a la primera entrada la llamamos X y a la segunda entrada Y, podemos observar que, a partir de la tabla de verdad de un sumador medio (véase p. 513), la salida (S) suma (el resultado) corresponde a la expresión booleana: $S = \bar{X}.Y + X.\bar{Y}$. Utilizando la ley de Morgan, obtenemos esta simplificación: $S = \overline{X.Y}.(X + Y)$. La salida llevo (C) es sencillamente: $C = X.Y$.

En aritmética binaria hay, de hecho, tres dígitos a sumar en una columna cualquiera de la suma de adición. Además de los dos dígitos a sumar, hay también un llevo procedente de la columna previa que incluir. Para poder reproducir el proceso de la suma normal debemos diseñar un circuito con tres entradas y dos salidas. Si al llevo de la columna anterior lo llamamos P, entonces la tabla de verdad para un sumador completo sería:

ENTRADAS			SALIDAS	
P	X	Y	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Tomando los casos en los que S = 1, la expresión booleana para S a partir de la tabla de verdad sería la siguiente:

$$S = \bar{P}.\bar{X}.Y + \bar{P}.X.\bar{Y} + P.\bar{X}.\bar{Y} + P.X.Y$$

Utilizando las reglas que hemos aprendido, podemos simplificar esta expresión:

$$S = \bar{P}.(\bar{X}.Y + X.\bar{Y}) + P.(\bar{X}.\bar{Y} + X.Y)$$

(propiedad distributiva)

$$S = \bar{P}.(\bar{X}.Y + X.\bar{Y}) + P.(\overline{\bar{X}.Y + X.\bar{Y}})$$

(ley de Morgan)

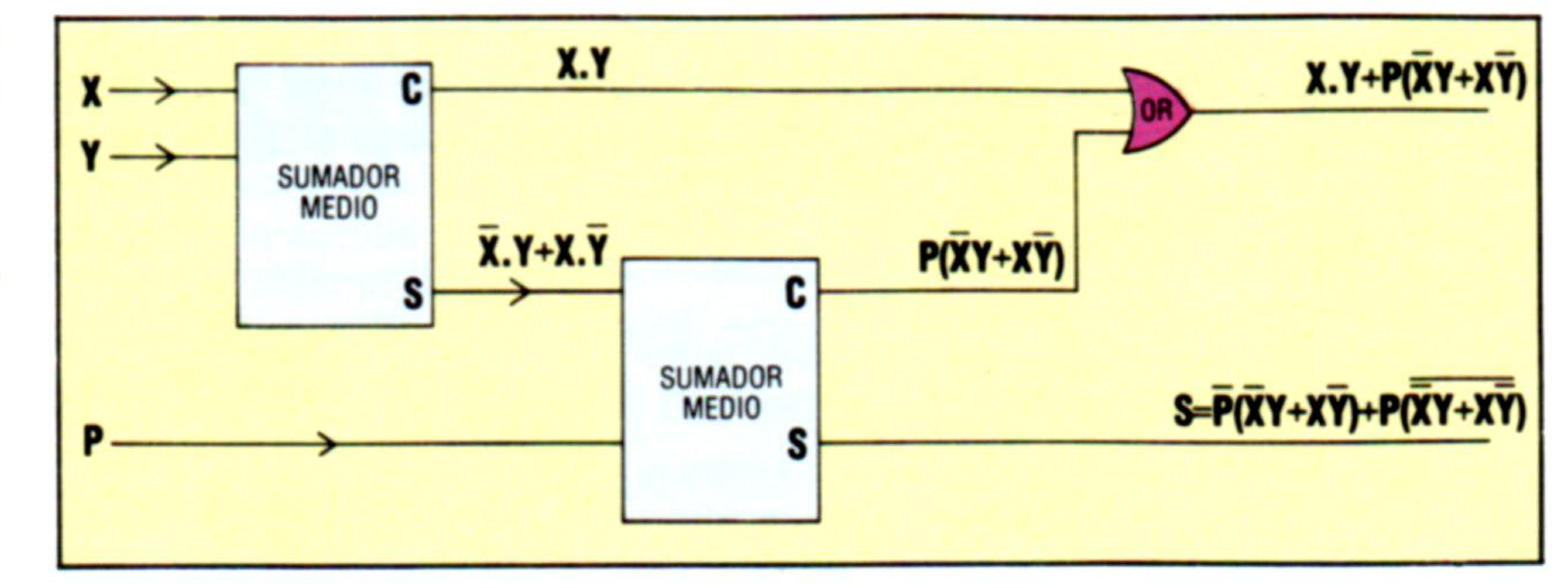
Del mismo modo, podemos formar una expresión para C. A partir de la tabla de verdad:

$$C = \bar{P}.X.Y + P.\bar{X}.Y + P.X.\bar{Y} + P.X.Y$$
$$C = X.Y.(\bar{P} + P) + P.(\bar{X}.Y + X.\bar{Y})$$

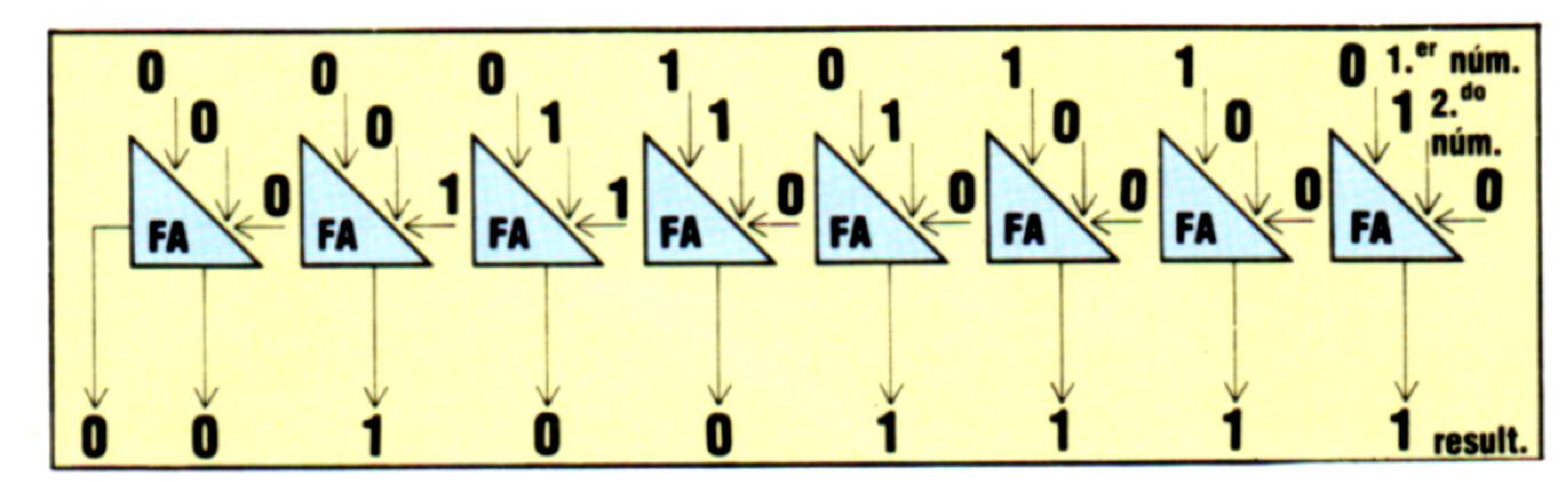
(propiedad distributiva)

$$C = X.Y + P.(\bar{X}.Y + X.\bar{Y}) \qquad (\bar{P} + P = 1)$$

Observe que $\bar{X}.Y + X.\bar{Y}$ es la salida suma de un circuito sumador medio. Por consiguiente, se puede diseñar un circuito sumador completo a partir de dos sumadores medios.



Veamos cómo una serie de ocho sumadores completos se combinan dentro de la ALU para efectuar la adición binaria de dos números de ocho bits.



EJERCICIO 3

1) Simplificar estas expresiones:

a) $A.(\bar{A} + \bar{B})$

b) $X + Y.(X + Y) + X.(\bar{X} + Y)$

c) $P.Q + \bar{P}.Q + \bar{P}.\bar{Q}$

d) $\bar{X} + \overline{Y.Z} + \bar{Z}.Y$

2) Una alarma para automóvil posee un interruptor on/off e interruptores en las puertas delanteras. La alarma sonará si una de éstas o ambas se abrieran cuando el interruptor on/off estuviera colocado en *on*. Dibuje una tabla de verdad que muestre las tres entradas y la salida alarma. Utilice esa tabla de verdad para escribir una expresión booleana para el sonido de la alarma y dibuje un circuito lógico para el sistema de alarma.

3) La luz de un recibidor funciona desde un interruptor situado en la puerta, un interruptor situado al pie de la escalera o uno situado arriba de las escaleras. Diseñe un circuito lógico apropiado.

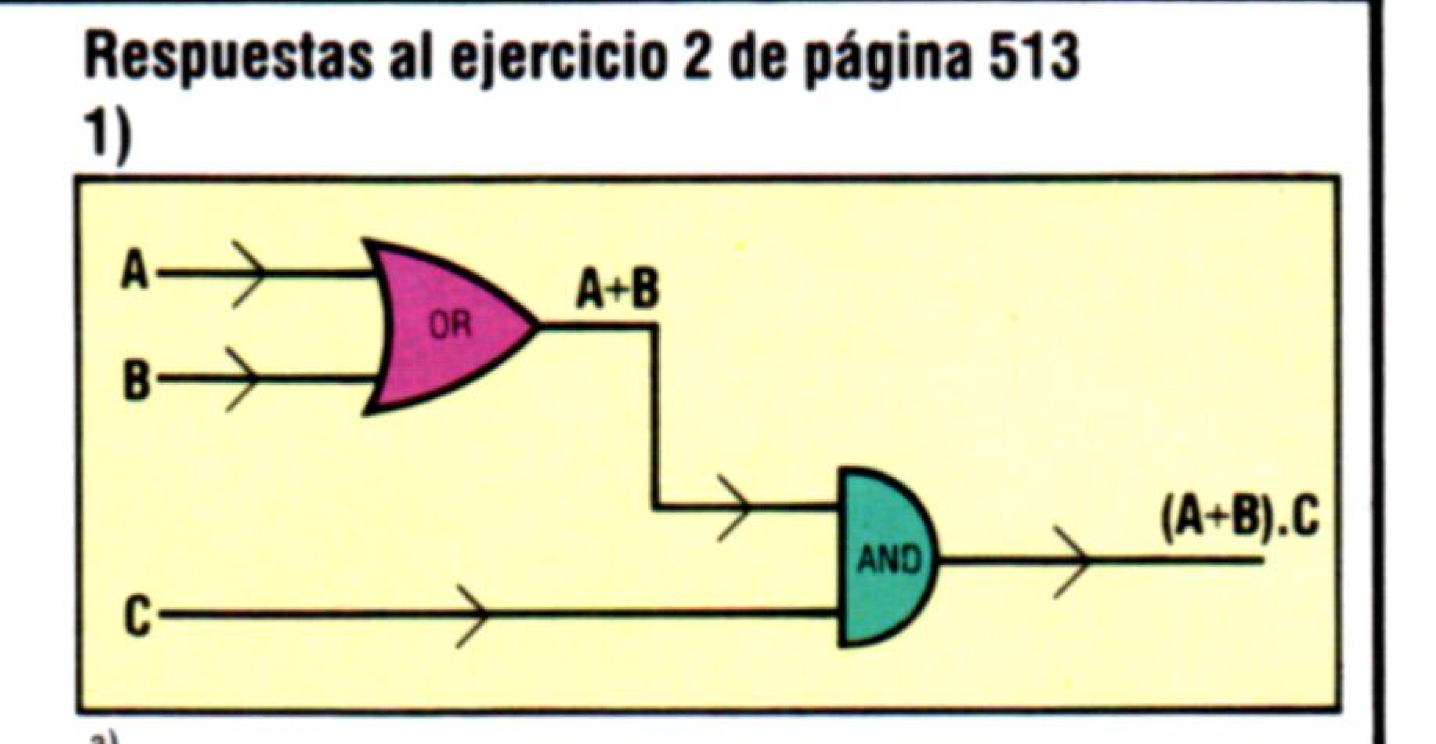
4) Acaban de abandonarle en una isla desierta con otras dos personas. Una de ellas siempre dice la verdad, la otra siempre miente. Si en algo aprecia su vida, ha de descubrir cuál de las dos es amiga de la verdad. Para ello hay varias preguntas que podría formularles a ambas. Confeccione tablas de verdad para investigar las posibles respuestas. He aquí un ejemplo:

"¿Dice usted siempre la verdad?"

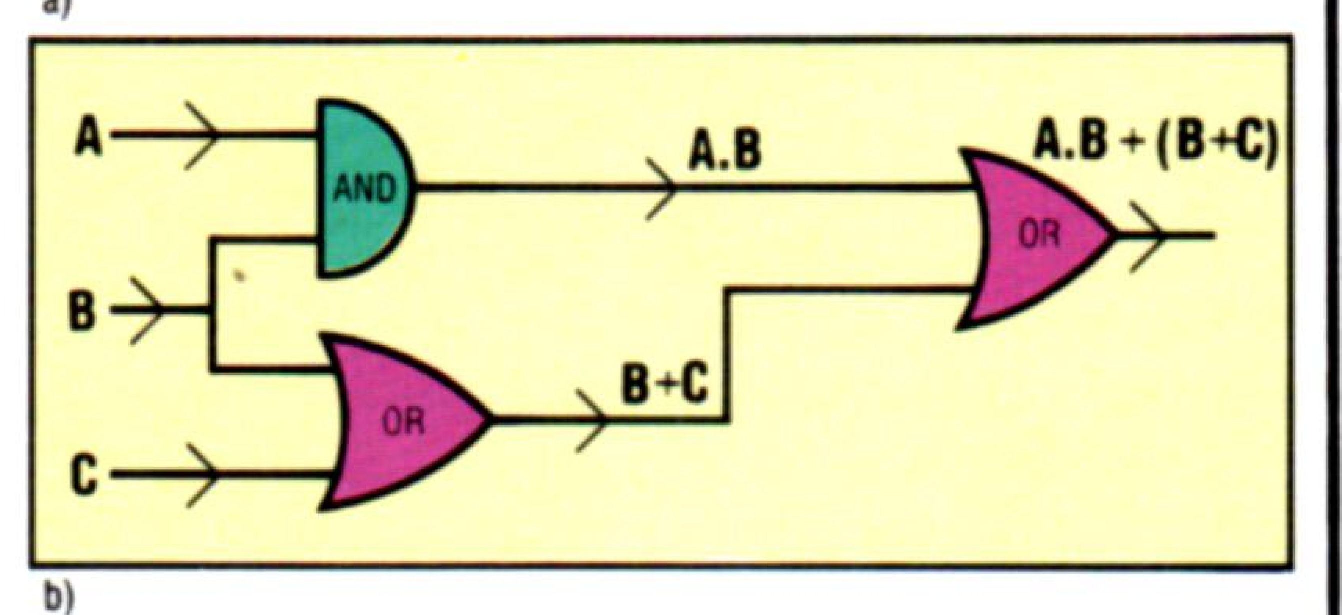
		POSIBLES RESPUESTAS	
		SI	NO
POSIBLE IDENTIDAD DE QUIEN CONTESTA	MIENTE	1	0
	DICE LA VERDAD	1	0

Respuestas al ejercicio 2 de página 513

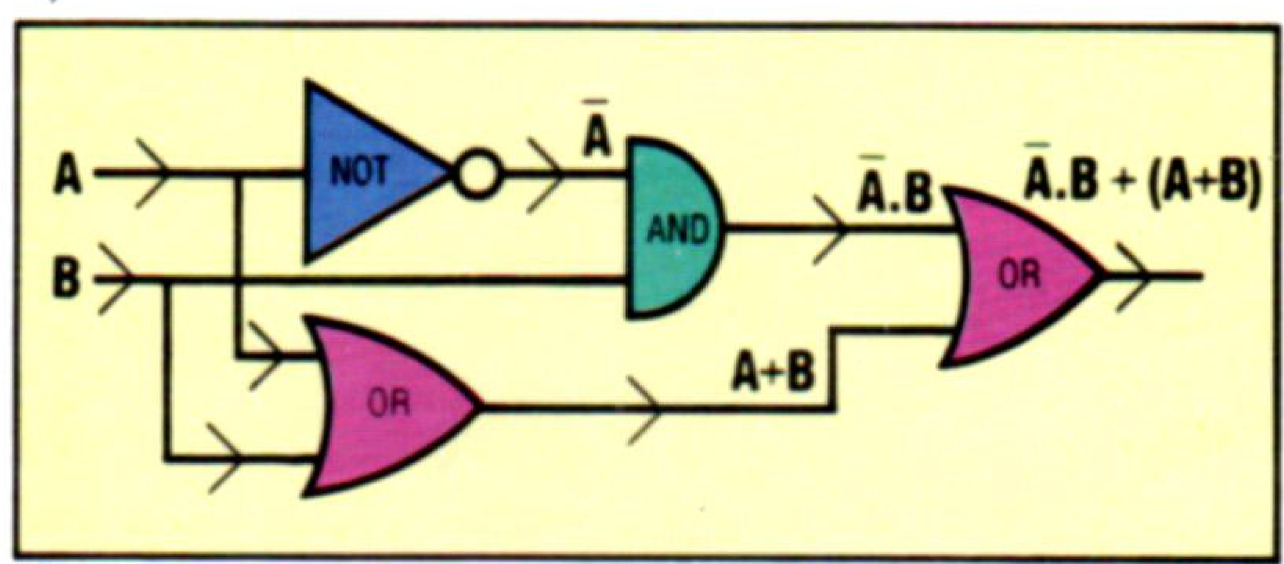
1)



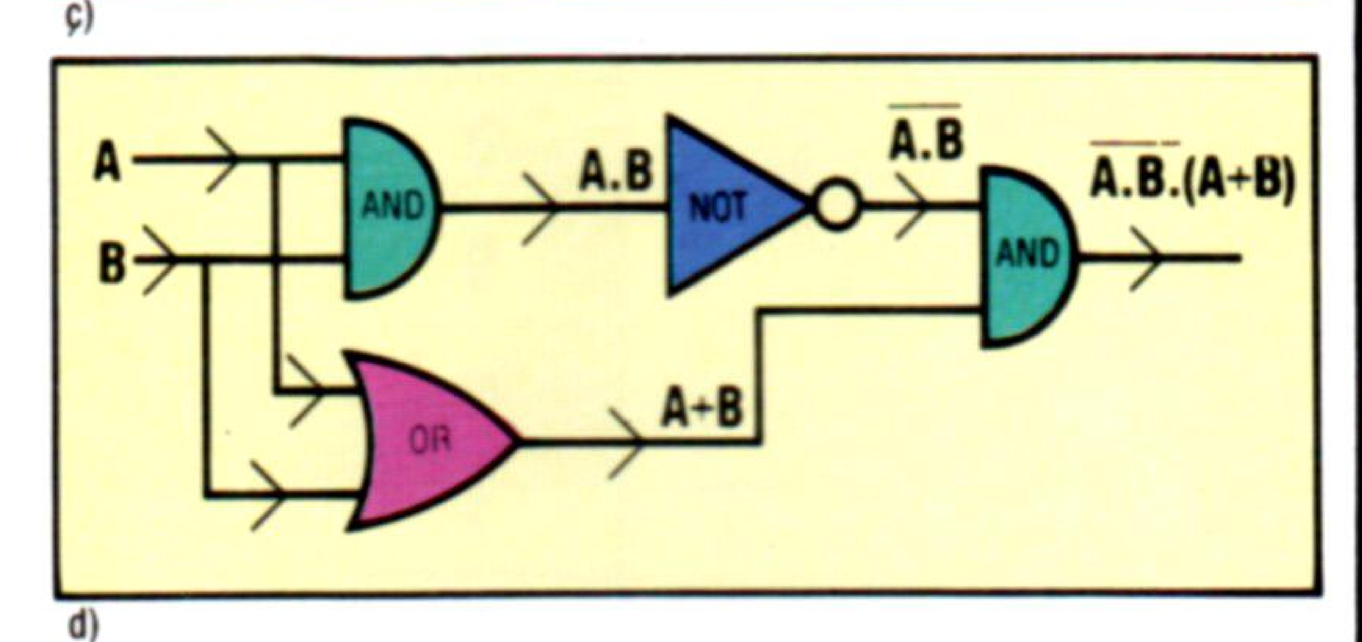
a)



b)



ç)



d)

2a) $C = A.B$
b) $S = \overline{A.B}\ (A + B)$
3a) $X = A + B).(B.C + C)$
b) $X = \overline{A.B} + \bar{B}$

Diagrama de bloques

No siempre ha de utilizarse este diagrama, pero en los casos más complejos evidencia la lógica del programa con toda claridad

El diagrama de bloques, llamado también *flowchart de programa*, es aquel que representa detalladamente las funciones de un programa y sus relaciones. Los procedimientos que llevan a la solución de un problema se componen de una sucesión de fases elementales en correcta secuencia. Este desarrollo se hace menos necesario a medida que el problema pierde complejidad, ya que en muchos casos se puede resolver simplemente con la aplicación de una fórmula, prescindiéndose, dada su síntesis, del empleo de un proceso. Pero en problemas complicados, resulta casi indispensable acudir a él para alcanzar el resultado.

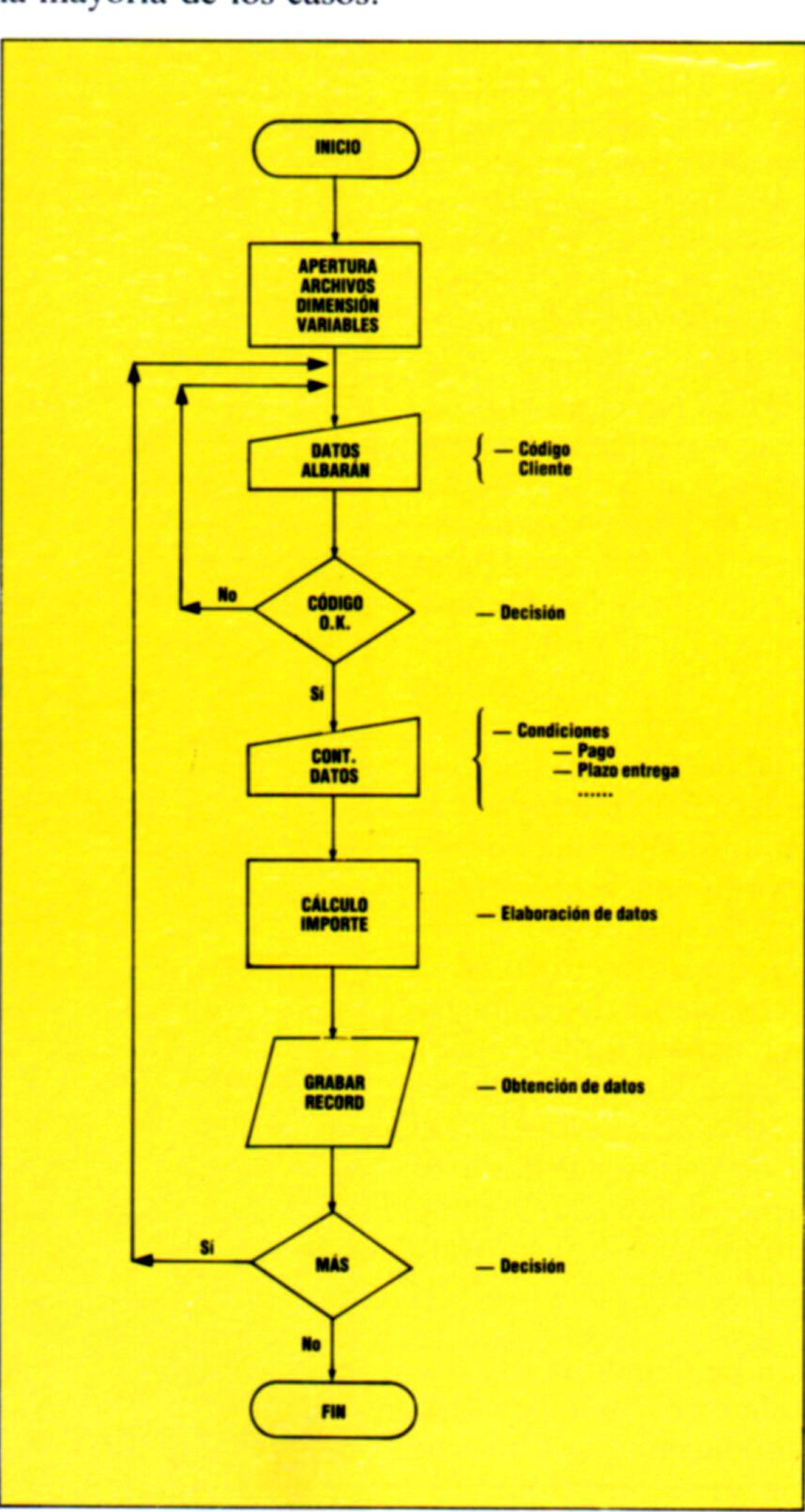
Así, definiendo las fases elementales de las diferentes alternativas de un problema y describiendo su secuencia correcta, puede llegarse a programar su solución. De igual manera, un diagrama de bloques proporciona una serie de informaciones diferentes. En primer lugar, sirve como documentación del programa, ya que se puede recurrir a él en caso de duda de la secuencia utilizada durante la programación o después de ella, y al mismo tiempo muestra de forma gráfica la solución del problema, a la que se llega tras seguir la secuencia de símbolos.

Un diagrama de bloques debe, asimismo, representar la lógica del programa, y en último lugar ha de verificar que se han tenido presentes todas aquellas variantes posibles que puedan influir en la resolución, sin dejar ningún punto al azar.

Tomando el ejemplo del capítulo anterior sobre la automatización de la gestión de una empresa figurada, que sirvió para ver la utilización del diagrama de flujo, representemos el que nos ocupa actualmente y que será aquel que se entregará al programador para que pueda llevar a cabo su labor.

1. En primer lugar, se hará la apertura de los archivos con los que se vaya a trabajar (clientes, artículos, etc.) y se dimensionarán una serie de variables destinadas a contener los datos que se utilizarán a lo largo del desarrollo del programa (tablas, contadores, acumuladores, etc.). Habrá que desarrollar después lo que se va a convertir en la secuencia repetitiva del programa.
2. Comienza con la entrada por teclado del número de código del cliente a cuyo cargo corre el albarán.
3. Debe hacerse en este punto una comprobación (de forma automática) consistente en comparar el mencionado código con el registro correspondiente, dando opción así a dos posibles salidas: *a*) en caso de que el código sea erróneo, la salida supondrá volver de nuevo al punto *b*, donde se introducirá nuevamente el código del cliente; y *b*) si se comprueba que el código es correcto, el proceso pasará secuencialmente a las demás fases.
4. Se da aquí entrada, también por teclado, al resto de datos que se precisen para poder actualizar el registro correspondiente y para usarlos en las pertinentes operaciones.
5. Supone este paso la elaboración y obtención de resultados. En él se realizarán las operaciones necesarias, partiendo de los datos disponibles.
6. Al llegar aquí, los datos modificados se incluyen en el registro y éste se graba nuevamente en el fichero, conservándolo así debidamente actualizado.
7. Por último, se ofrece la posibilidad de repetir el proceso tantas veces como se quiera. Si se continúa, la secuencia retornará al punto *b*; en caso contrario, finaliza el programa.

Con este ejemplo, cuya solución gráfica se adjunta, queda patente la utilidad del diagrama de bloques, del que un buen programa no debería prescindir en la mayoría de los casos.



Hasta el límite

La tercera versión del "best-seller" Spectrum (50 000 unidades vendidas en España en tan sólo 11 meses) presenta particularidades dignas de ser analizadas más de cerca

Como elemento de hardware, la peculiaridad más notable del Spectrum es el teclado. Si bien podemos afirmar que es del tipo QWERTY, hasta aquí llega todo su parecido con una máquina de escribir. Cada una de las 40 teclas es parte de una membrana que confiere a las teclas cierto recorrido: al "tacto", estas teclas parecen de esponja.

El Spectrum posee un conector de buses del sistema, que permite al usuario conectar simultáneamente la impresora ZX (diseñada originalmente para el ZX81), la interface 1 ZX y la interface 2 ZX. También hay conectores MIC y EAR, que permiten el almacenamiento en cassette de programas. El procedimiento para cargar programas no es muy satisfactorio. Si bien al cargar y guardar correctamente aparecen sobre el borde de la pantalla unas franjas azules y amarillas que así lo indican, para poder guardar un programa uno debe primero desconectar el conductor EAR. Igualmente incómoda es la carencia de un botón Reset en el ordenador, lo que significa que cada vez que se produce una rotura del sistema se ha de quitar el conector de alimentación eléctrica, lo que con el tiempo podría debilitar las conexiones. Por suerte, algunas pequeñas firmas independientes producen dispositivos accesorios que incorporan tanto un interruptor para guardar o cargar *(save or load)* como un interruptor para restauración *(reset)*.

Esto es sólo un ejemplo de la forma en que Sinclair Research parecería ceder en su interés por la máquina. Puede que sea, no obstante, una política deliberada de la empresa, porque mientras disfruta de los beneficios de las ventas de su ordenador, Sinclair no ahorra esfuerzos en un proyecto como el QL. Aun así, la empresa ha desarrollado los microdrives ZX y la unidad relacionada interface 1 ZX, que proporcionaron el potencial de almacenamiento de apoyo de 680 Kbytes, una puerta RS232 y la idea de conectar en redes de área local hasta 64 unidades Spectrum. Asimismo, se introdujo la unidad interface 2 ZX, que permite acceso al software basado en RAM y conectar dos palancas de mando.

Sinclair Research también ha dejado la producción de software en manos de otros, al haber dado acogida a ciertos paquetes producidos por firmas de software independientes con su autorización.

Evolución del Spectrum
Desde que se introdujo en el mercado británico, en 1982, hasta su eclipse parcial por el QL, a comienzos de 1984, del ZX Spectrum de Sinclair se vendió más de un millón de unidades y pasó por tres versiones en su tablero original (o sea, tablero de circuito impreso que sostiene todos los componentes principales). La primera versión se utilizó sólo para las primeras 60 000 unidades que se vendieron, de modo que no es común. Las dos últimas versiones difieren en dos aspectos primordiales. Primero, en la segunda versión se podía "afinar" el sistema de circuitos de salida de video mediante los dos condensadores de equilibrio y las dos resistencias variables que vemos en la fotografía. En segundo lugar, la "modificación temporal" del microprocesador perteneciente a la segunda versión se logró formalizar cuando se introdujo la tercera versión. El disipador de calor está en un lugar diferente porque el chip regulador de voltaje se ha colocado más cerca del conector para entrada de corriente

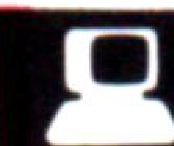
Resistencias variables

Condensadores de equilibrio

Cristal del reloj principal
Funciona a 14 MHz

Disposición lógica no comprometida
Este chip de 40 patillas sustituye a una gran variedad de chips lógicos, controla las operaciones de entrada-salida, incluyendo la generación de una señal de video compuesta, que después se modula con una frecuencia de radio, y controla los interruptores de la CPU

Conectores del te



Cristal reloj video en color
Funciona a 4,4336 MHz

Unidad Central de Proceso Z80A
Observe la "modificación temporal" para los tableros originales de la segunda versión, haciendo un puente entre las patillas 11 y 30 mediante un transistor. Esta modificación tiene el efecto de permitir que se seleccione la ULA sólo cuando la línea de dirección cero y la IORQ (*Input/Output Request:* solicitud de entrada-salida) están las dos bajas

Regulador de voltaje

do

Ian McKinnell

SINCLAIR SPECTRUM

DIMENSIONES
233 × 144 × 30 mm

CPU
Z80A

CAPACIDAD DE MEMORIA Y VELOCIDAD
16 K y 48 K
3,5 MHz

CARACTERISTICAS DE LA PANTALLA
La pantalla está dividida en 24 líneas de 32 caracteres. Gráficos con mapas de bits a una resolución de 256 × 192. Dieciséis caracteres ya programados para gráficos de bloques más 21 caracteres para gráficos definibles por el usuario. Ocho colores, flash y dos niveles de brillo. Color independiente para el borde

INTERFACES Y PUERTAS
Conector para bus del sistema. Conectores para almacenamiento en cassette y televisor

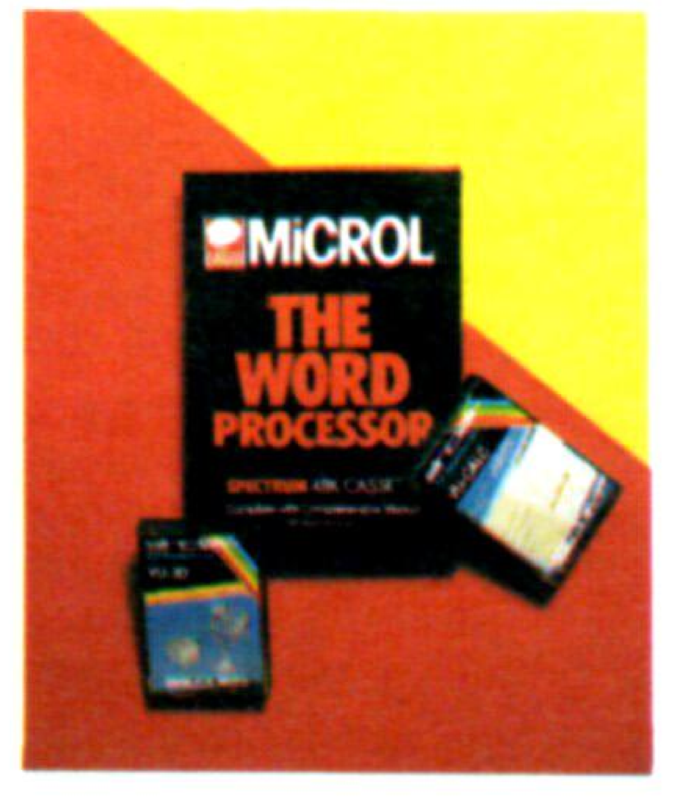
LENGUAJES DISPONIBLES
BASIC y lenguaje ensamblador Z80. La ampliación mediante software permite utilizar FORTH, LOGO, Micro-PROLOG y otros

TECLADO
Teclado de 40 teclas móviles ASCII tipo membrana. El diseño permite reclamar hasta ocho funciones desde una sola tecla a través de una serie de teclas Shift (cambio)

DOCUMENTACION
El ordenador viene completo con un pequeño manual de introducción y un manual más amplio con una guía para operar el micro y orientación acerca del BASIC de Sinclair

PUNTOS FUERTES
En su versión de 48 K, el Spectrum ofrece una configuración ideal para experimentación del recién iniciado. Goza de gran apoyo de software y hardware producido por fabricantes independientes

PUNTOS DEBILES
El teclado, aunque diseñado con gran inteligencia, no ofrece las facilidades del tipo máquina de escribir. La visualización en pantalla está configurada de una forma no estandarizada, y puede plantear problemas a las personas sin experiencia



El archivo del Spectrum
Además del elevado número de paquetes de juegos existentes para el Spectrum, también existe en el mercado una estimable cantidad de software para gestión y administración doméstica, gran parte del cual se vende a un precio muy asequible. Con la introducción del microdrive ZX, no existe razón alguna por la cual el Spectrum no pueda aplicarse al mantenimiento de grandes bases de datos, hojas electrónicas y tratamiento de texto.
Psion, uno de los colaboradores más próximos de Sinclair en la producción de software, ha sido el responsable del nacimiento de gran parte de este material. Entre éste destaca la producción de la exitosa serie de paquetes VU, que incluye el VU-CALC, VU-File y VU-3D. Y, a pesar de las deficiencias del teclado del Spectrum, Microl ha producido *The word processor* (procesador de textos), que ofrece espacio de almacenamiento equivalente a 10 páginas A4 de texto, y la mayoría de las facilidades que normalmente ofrecen los paquetes más completos para edición de textos (incluyendo la posibilidad de mezclar archivos)

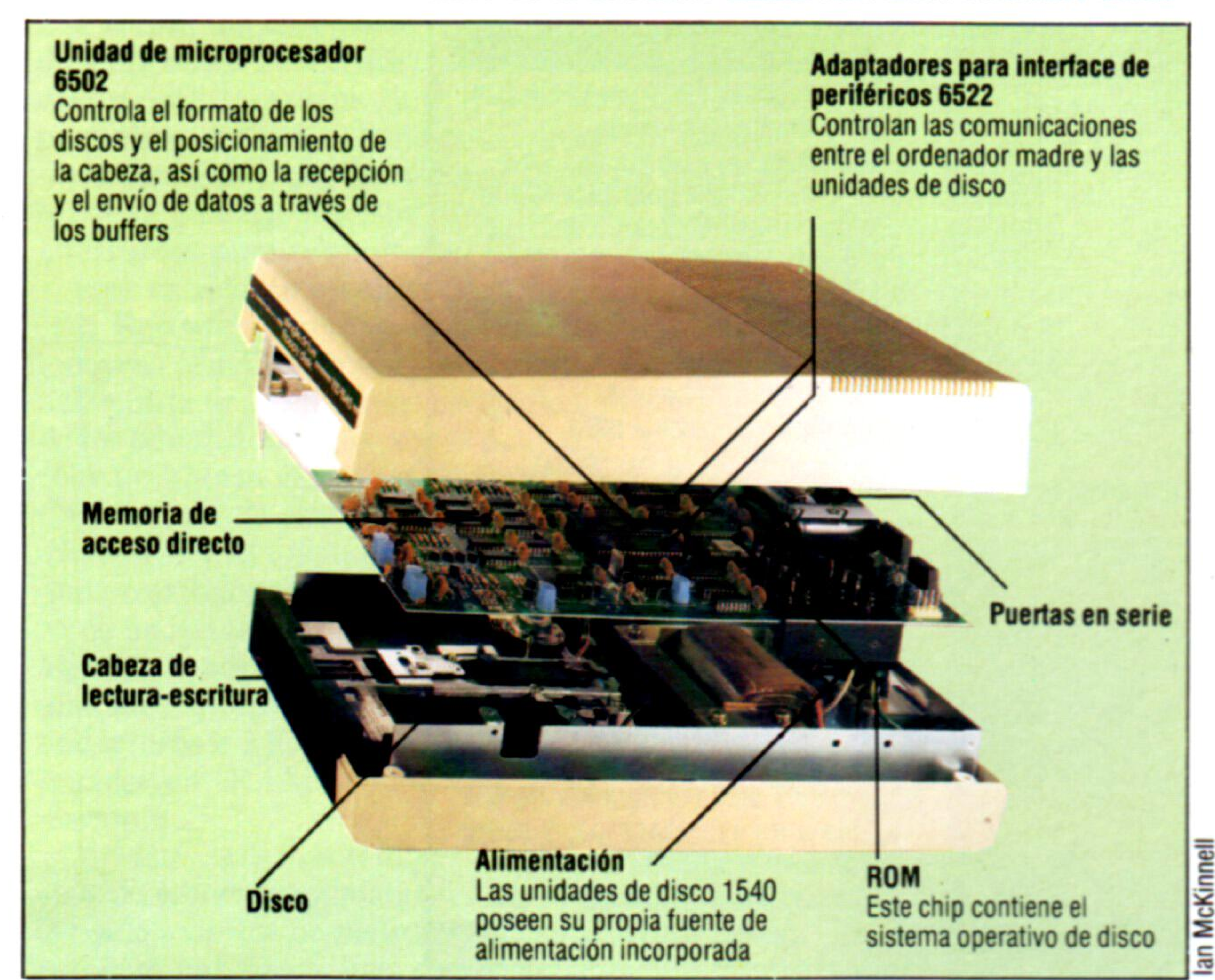
Unidad emancipada

A diferencia de otros sistemas de disco, las unidades Commodore son "inteligentes", ya que cuentan con su propio microprocesador y su RAM

El principal problema de las unidades de disco inteligentes es que su fabricación es cara. Después de la introducción del ordenador personal Vic-20, Commodore lanzó una versión económica de la unidad de disco PET, denominada Vic-1540. El Commodore 64 incorpora facilidades similares a las del Vic-20 para acceder a la 1540, pero diferencias menores hicieron que fuera necesario efectuar una POKE antes de utilizar la unidad y otra POKE al terminar. Pero tan engorroso procedimiento ya no es necesario, porque Commodore realizó algunas modificaciones en el DOS para rectificar el defecto y volvió a lanzar la 1540 como la 1541. Esta versión más reciente es totalmente compatible tanto con el Vic-20 como con el 64. Por razones de simplicidad nos referiremos a ambas unidades como 1541, porque no existe ninguna diferencia en cuanto a la forma en que se utilizan.

La 1541 se controla a través de un microprocesador 6502, dos VIA (*Versatile Interface Adaptors*: adaptadores versátiles para interface) 6522, dos Kbytes de RAM y ocho Kbytes de ROM, que contiene el DOS. El DOS que se proporciona es muy potente y permite programar complicadas rutinas para crear y manipular archivos de programas (PRG), archivos de datos secuenciales (SEQ) y archivos de acceso directo, todos con refinados procedimientos para verificación de errores. El control del ordenador se ejerce a través de una versión en serie de la interface IEEE488. Esta interface admite las mismas órdenes que su equivalente en paralelo más poderoso (a través del cual se controlan los otros periféricos Commodore) y permite "conectar en cadena margarita" periféricos dotados de IEEE488 en serie, de modo que, por ejemplo, una unidad de disco puede dar salida a archivos hacia una impresora mientras el ordenador lleva a cabo otra tarea. Esto se consigue mediante la utilización de órdenes con los números de archivos lógicos y números de los dispositivos correspondientes.

La unidad de disco Commodore
Disponible tanto para el Vic-20 como para el Commodore 64, al ser una configuración estándar del más reciente ordenador portátil SX-64, la unidad de disco 1540 de Commodore es una unidad inteligente que no exige nada ni a la CPU ni a la memoria del ordenador al que se acopla. De hecho, posee su propio procesador, el MOS Technology 6502, que alimenta a los mismos ordenadores



Ian McKinnell

El formato de los diskettes es de 35 pistas por cara; las pistas están divididas en sectores, desde 21 en la pista exterior hasta 17 en la más interior. Cada sector contiene un bloque de 256 bytes de datos de archivo, más datos de sincronización, identificación y suma de control. Cada diskette almacena 683 bloques, de los cuales 664 están disponibles para el usuario. Esto otorga una capacidad máxima de aproximadamente 170 Kbytes, según el tipo de archivos que se almacene. El DOS administra la distribución de datos en un diskette manteniendo un mapa de disponibilidad de bloques (*Block Availability Map*: BAM) y un directorio. El BAM se almacena en la pista 18, sector 0, y consta de 144 bytes que indican qué bloques están en uso y cuáles están libres para el almacenamiento. El directorio comienza en el sector 1, pista 18, y es una lista de un máximo de 144 archivos, con el nombre de cada uno, que contiene información específica relativa al tipo de archivo y de cuántos bloques consta. Tanto el BAM como el directorio se van actualizando a medida que se escriben datos en el diskette o que se eliminan del mismo.

A pesar de su elevado precio de venta, la flexibilidad del sistema de unidad de disco Commodore ofrece una magnífica relación calidad-precio. Las posibilidades de su fiable almacenamiento masivo (que se puede incorporar a un sistema de administración de periféricos y no se inmiscuye ni en la memoria del ordenador ni en el tiempo del procesador) justifican el gasto. No obstante, es lamentable que, debido a la interface en serie, la 1541 sea de operación comparativamente lenta. En el manual para el usuario, Commodore ni siquiera menciona la velocidad de transferencia de datos ni el tiempo de acceso promedio. Aunque es con creces 50 veces más rápido que el almacenamiento de cassette, es probable que el 1541 sea el de más lento acceso de todos los sistemas de disco más populares. Si se necesita una gran capacidad de almacenamiento, se puede adquirir una interface que se conecta en la puerta en serie y que imita a la puerta en paralelo de que disponen los ordenadores PET. Esto no acelera especialmente las operaciones, pero sí permite que usted conecte toda la gama completa de periféricos PET a un Vic-20 o un Commodore 64.



El manejo del disco...

El DOS de Commodore admite una amplia gama de órdenes diseñadas para que el usuario construya complicados programas para manipular archivos de acceso directo, así como el programa normal y la manipulación de archivos de datos. Estas órdenes se utilizan tal como vemos más abajo. En todos los casos, 8 es el número identificador de dispositivo de la unidad de disco.

SAVE

Crea archivos de programas (PRG) con nombre (hasta 16 caracteres) que pueden ser programas o datos secuenciales. El formato es el siguiente:

SAVE"NOMBRE ARCHIVO",8

LOAD

Se construye así:

LOAD"NOMBRE ARCHIVO",8

Esta orden copia el archivo PRG especificado en RAM desde la parte inferior de la memoria para usuario hacia arriba. La orden

LOAD"NOMBRE ARCHIVO",8,1

vuelve a copiar el archivo especificado en las posiciones de memoria de las cuales se tomó para guardar (SAVE) originalmente.

LOAD"$",8

copia el directorio del disco en la memoria para el usuario. Luego se puede LISTar como un programa en BASIC y contiene lo siguiente:

Nombre del disco
Identificador de disco de 2 caracteres
Hasta 144 nombres de archivos
Tipo (PRG o SEQ) de cada archivo
Longitud en bloques de cada archivo
Número de bloques libres disponibles

VERIFY

Se construye así:

VERIFY"NOMBRE ARCHIVO",8

y compara el archivo especificado con el archivo que contiene la memoria para el usuario y genera un mensaje de error en el caso de que sean diferentes. Se utiliza para verificar que los archivos se hayan guardado (SAVE) correctamente.

OPEN

Establece un canal de comunicación exclusivo que se identifica mediante un "número de archivo lógico" (*Logic File Number: LFN*) dentro del intervalo [1, 255]. Se pueden abrir (OPEN) simultáneamente hasta 10 LFN. OPEN también establece una "dirección secundaria" (*Secondary Adress: SA*), que determina la forma en que se comportará el dispositivo al cual se accede. La única dirección secundaria de la unidad de disco es 15, que da acceso al canal de prioridad *command* (orden). A OPEN se le da entrada del siguiente modo:

OPEN LFN,8,SA

CLOSE

Asume el formato:

CLOSE LFN

Termina el archivo lógico especificado. Los archivos lógicos siempre se deben cerrar (CLOSE) cuando ya no se los necesita.

PRINT#, INPUT# y GET#

PRINT# funciona de modo similar a PRINT con la excepción de que a los datos se les da salida, como un archivo SEQ, al archivo lógico OPEN especificado y no a la pantalla. Se construye así:

PRINT#LFN,"DATOS", o bien
PRINT#LFN,A$,B$,...

Del mismo modo, INPUT# y GET# leen archivos SEQ. INPUT# recupera datos en cadena pero sólo es eficaz si las cadenas almacenadas están separadas mediante punto y coma o sólo comas, de lo contrario INPUT# tratará los datos como una cadena larga. GET# recupera los datos a un byte por vez, incluyendo las comas, o los punto y coma. Ésta es más útil cuando no se conoce el contenido de un archivo y éste no está separado. Los siguientes ejemplos ilustran los formatos:

INPUT#LFN, A$, B$...
GET#LFN, A$,B$...,

Cuando PRINT# se utiliza junto con un archivo lógico abierto (OPEN) al canal de órdenes (p. ej., OPENLFN,8,15) de esta manera:

PRINT#LFN,8,15,"serie de órdenes"

se transforma en la orden para manipulación de disco más poderosa de las disponibles. Las series de órdenes se utilizan para ejecutar órdenes para mantenimiento de disco y avanzadas órdenes para archivos de acceso directo (*relative*: REL).

...y su mantenimiento

Utilizadas conjuntamente con PRINT# u OPEN en el canal de órdenes, en el formato indicado, estas series de órdenes realizan las siguientes funciones:

NEW

Da formato y nombre al diskette
Construye el BAM y el directorio
Asigna el identificador de disco de 2 caracteres (DI)
Orden: "N:NOMBRE DISCO,DI"

INITIALISE

Verifica el BAM en RAM de disco con el BAM del disco
Orden: "I"

VALIDATE

Elimina los bloques destinados mediante avanzadas órdenes REL no retenidas en el directorio y archivos que no se han cerrado (CLOSE)
Escribe un BAM nuevo
Orden: "V"

RENAME

Modifica el listado del directorio de un archivo especificado
Orden: "R:NOMACTUAL = NOMANTERIOR"

SCRATCH

Borra archivos especificados del disco o directorio
"S:NOMARCH 1, NOMARCH 2,..."

COPY

Escribe una copia de un archivo en el mismo disco
Orden: "C:NOMDUP + NOMORIS"
Junta archivos SEQ y los escribe como un único archivo SEQ en el mismo disco. Conocida como orden "concatenante":
"C:CONNOMB = NOMB1,NOMB2,..."

Verificación de errores

En el panel frontal de la unidad de disco 1541 hay un LED (*Light Emitting Diode*; diodo emisor de luz) verde "encendido" y un LED rojo que indica el estado del disco, donde:

Encendido = Leyendo un disco o escribiendo en él
Apagado = Esperando instrucciones
Intermitente = El DOS ha detectado un error

Para descubrir la naturaleza del error se debe leer el canal de error del DOS. El siguiente programa imprime los códigos de error generados por el DOS. En el manual para el usuario de la unidad de disco se proporciona una lista de los códigos de error y sus significados.

```
10 REM**VERIFICACION
   ERROR DE DISCO**
20 OPEN15,8,15
30 INPUT#15,EN,EM$,ET,
   ES
40 PRINTCHR$(147)
50 PRINT"N.° ERROR"EN
60 PRINTEM$
70 PRINT"PISTA"ET
80 PRINT"SECTOR"ES
90 CLOSE15:END
```

El ABC del BBC

Continuando con nuestro estudio del BASIC de los ordenadores personales más populares, ahora examinaremos las particularidades de la versión que incorpora el BBC Micro, una de las más elogiadas

Entre líneas

No olvide numerar las líneas de su programa en múltiplos de diez. Hasta los mejores programadores tienen que hacer inserciones en sus programas de cuando en cuando...

La crítica que suele suscitar con más frecuencia el BASIC es que se trata de un lenguaje sin estructurar que favorece (o, al menos, que no hace nada por controlar) malos hábitos de programación en el principiante, en particular la solución "rápida y sucia" de los problemas, lo que lleva, por ejemplo, a la utilización indisciplinada de GOTO.

El empleo de ELSE con la sentencia IF...THEN puede eliminar la utilización más común de GOTO, al permitir que en la misma sentencia se traten tanto los casos verdaderos de una condición como los falsos. Por ejemplo, estas líneas:

```
1500 IF TEST > 0 THEN GOTO 1800
1600 PRINT "VALOR FUERA DE ESCALA"
1700 GOTO 1900
1800 PRINT "NO HAY PROBLEMA"
1900 NEXT L
```

se pueden sustituir por:

```
1500 IF TEST > 0 THEN PRINT "NO HAY
     PROBLEMA" ELSE PRINT "VALOR FUERA DE
     ESCALA"
1900 NEXT L
```

GOSUB suele tomar como argumento un número de línea, lo que ofrece dos inconvenientes; en primer lugar, GOSUB 1000, por ejemplo, no proporciona ninguna pista acerca del objetivo de la subrutina de la línea 1000; en segundo lugar, la especificación de números de línea hace que sea muy difícil volver a numerar el programa o mezclarlo.

GOSUB, como GOTO, es de ejecución relativamente lenta, porque cada vez que se obedezca la instrucción se ha de buscar en el programa la línea especificada.

Las funciones y los procedimientos del BASIC del BBC responden a estas objeciones. Ambas son bloques de códigos a modo de subrutinas, pero se las llama por un nombre en vez de por el número de línea, de manera que pueden estar autodocumentadas o al menos tener un significado en el listado, y no necesitan verse afectadas por subsiguientes renumeraciones o mezclas. Además, las llamadas a funciones y procedimientos por lo general se ejecutan con más rapidez que las órdenes GOSUB y GOTO.

Los procedimientos y las funciones empiezan con DEF PROC o DEF FN, seguido de un nombre, y casi siempre (pero no necesariamente) una lista de parámetros. Por ejemplo:

```
1200 DEF FNcalc(a,b,c) = (a−b)* c/100 y
2500 DEF PROCoperac (w,x$,y$,z)
```

La definición utilizará estos parámetros como si fueran variables del programa. Sin embargo, cuando el programa llama a la función o el procedimiento, los parámetros, o variables ficticias, se pueden sustituir por cualquier número de variable o expresión literal del mismo tipo de datos que el parámetro original. Por ejemplo:

```
250 resultado = FNcalc (precio, costo, 12) o
545 PROCoperar (6, nombres$, "pérez",
    matriz (12))
```

Los valores de los parámetros se emplean entonces en la definición en lugar de las variables ficticias. Observe que una función se puede utilizar en una expresión como si fuera una variable o una cantidad aritmética, mientras que una llamada a procedimiento se emplea como si fuera una orden en BASIC. La orden LOCAL impide que se produzca el error de una subrutina común:

```
100 FOR K = 1 TO 10:GOSUB 500:NEXT K:END
500 FOR K = 1 TO 5:PRINT"****";NEXT:RETURN
```

Aquí la variable K se utiliza como el contador del bucle de la línea 100 del programa principal y otra vez en la subrutina de la línea 500, un descuido que afectará a la ejecución pero que puede resultar difícil de evitar (o de rastrear). Pero en un procedimiento BBC, este peligro se puede evitar:

```
100 FOR K = 1 to 10:PROCestrellas:NEXT:END
500 DEF PROCestrellas
520 LOCAL K
540 FOR K = 1 TO 5:PRINT"****":NEXT
560 ENDPROC
```

La orden LOCAL significa que entre las líneas 500 y 560 la variable K es una nueva variable, independiente de la variable K de cualquier otro lugar del programa, y que no tiene incidencia sobre cualquier otro valor de K. (Observe que PROCestrellas es un procedimiento sin parámetros.)

REPEAT...UNTIL es una estructura de bucle en la cual la iteración continúa hasta que la expresión condicionada que sigue a la palabra UNTIL sea verdadera; el control pasa entonces a la sentencia que le sigue. Por ejemplo:

```
200 DATA 12,234,31,45,65,0,76,81
250 REPEAT
300 READ número:suma = suma + número
350 UNTIL número = 0
400 PRINT "La suma es";suma
```



Esto resulta mucho más legible y es mucho menos susceptible de error que un bucle GOTO o un bucle FOR...NEXT ficticio.

El BASIC del BBC posee las valiosas ayudas para depuración que son TRACE, ON ERROR..., y ERL. TRACE hace que se visualicen en la pantalla los números de línea del programa a medida que se van ejecutando; ON ERROR GOTO (o GOSUB) significa que cualquier error que normalmente sería fatal (incluyendo el pulsar la tecla ESCAPE) durante la ejecución del programa, haría que el control pasara a una rutina para manipulación de errores definida por el usuario (como un volcado de todos los valores de las variables). ERL es una variable del sistema que contiene el número de la línea en la cual se ha producido el error.

El BBC posee una riqueza de ampliaciones para el BASIC exclusivas, como las llamadas al sistema operativo, la orden VDU, las diversas variables del sistema y el ensamblador, que son en sí mismas suficientemente complejas como para ser objeto de comentarios sueltos que ofreceremos más adelante.

Los gráficos BBC

Las órdenes del BASIC del BBC que están directamente relacionadas con los gráficos son:

MODE
Selecciona la modalidad visualización del ordenador con MODE N, donde N = de 0 a 7:

Modalid.	Gráficos	Colores	Texto
0	640×256	2	80×32
1	320×256	4	40×32
2	160×256	16	20×32
3	sólo texto	2	80×25
4	320×256	2	40×32
5	160×256	4	20×32
6	sólo texto	2	40×25
7	Teletexto		40×25

Los ordenadores BBC Modelo A sólo pueden acceder a las modalidades 4, 5, 6 y 7; el Modelo B tiene acceso a todas las modalidades. De la modalidad 0 hasta la 6, el usuario puede modificar el juego de caracteres mediante la orden VDU. Los caracteres de teletexto de la modalidad 7 son fijos y no corresponden al código ASCII estándar.

COLOUR
Establece uno de los 16 colores para texto y fondo según la modalidad seleccionada con:

COLOUR N
donde N va de 0 a 15 para los colores de texto y de 128 a 143 para los colores del fondo. Los colores establecidos por cada color de N no son constantes de una modalidad a otra. En la guía para el usuario se ofrecen listas de los valores de N en relación a los colores para cada una de las modalidades.

VDU
Ésta es una orden sumamente útil. VDU A equivale a PRINT CHR$(A). De la misma manera, VDU A,B,C produce los mismos efectos que PRINT CHR$(A);CHR$(B);CHR$(C). Esto significa que las muchas y complicadas rutinas para texto y gráficos bajo el control de los 32 códigos CHR$, que duplican los efectos de la mayoría de las órdenes de BASIC relacionadas con los gráficos, se pueden construir con un pequeño número de órdenes VDU.

CLG
Limpia la superficie para gráficos de la pantalla corriente y desplaza el cursor a su posición "base", en el extremo inferior izquierdo de la pantalla.

CLS
Limpia la superficie para texto de la pantalla y desplaza el cursor para texto hasta su posición "base", arriba a la izquierda. También se borrará todo gráfico que hubiera en la pantalla.

DRAW
Dibuja líneas en la pantalla en las modalidades 0, 1, 2, 4 y 5. Se construye así:

```
DRAW X,Y
```

El punto definido por las coordenadas X e Y es el final de la línea. El punto de partida puede ser o bien el punto final de la última línea trazada o un punto definido mediante una orden MOVE.

GCOL
Establece los colores corrientes del fondo y primer plano de los gráficos mediante:

```
GCOL N,M
```

donde N establece cómo se ha de emplear el color (de 0 a 4) y M define el color lógico utilizando los mismos principios que COLOUR. N posee los efectos siguientes:

0 — Trazar el color especificado por M
1 — OR color M con color actual
2 — AND color M con color actual
3 — OR-exclusivo color M con color actual
4 — Invertir color actual

MOVE
Posiciona el cursor para gráficos en un punto especificado mediante:

```
MOVE X,Y
```

Tiene el mismo efecto que DRAW pero sin trazar una línea.

PLOT
Se puede utilizar para muchas funciones de gráficos, incluyendo el trazado de puntos, líneas y triángulos. Se construye así:

```
PLOT K,X,Y
```

donde K define el tipo de gráfico a trazar (PLOT). K asume valores comprendidos en la escala 0-225 para especificar el tipo de líneas a dibujar y los colores que toman de acuerdo a las listas que se proporcionan en la guía para el usuario.

POINT
Proporciona el número relacionado con el color lógico de pantalla de la coordenada de pantalla especificada, por medio de:

```
VARNUM = POINT(X,Y)
```

donde VARNUM es una variable numérica.

Monitor de memoria

Al principio puede que la numeración hexadecimal le parezca un artilugio superfluo y rebuscado, pero poco a poco se irá percatando de que es una herramienta muy útil para tratar con la memoria

Llegados a este punto del estudio en torno al lenguaje máquina vale la pena insistir en el tema de la representación de los números. Nosotros estamos familiarizados con el sistema decimal de base 10 (denominado también *sistema denario*), que es el empleado en la vida diaria, y ya sabemos algo del sistema binario. Interesa recalcar que tanto el sistema decimal como el binario no son más que expresiones alternativas del mismo concepto: el número. El ser humano, salvo casos accidentales, tiene el mismo número de dedos en cada mano. Usted dice que el número es cinco, y alguna otra persona podría llamarlo *fünf*, o *cinq*, o *pente*; pero todas esas personas están aludiendo a la misma cantidad o al mismo número: lo único diferente es el sistema oral o escrito de representación. Son representaciones diferentes pero equivalentes. Hay una correspondencia de uno a uno entre todos los números expresados en castellano y todos los números expresados en cualquier otro idioma, pues todos estos sistemas tienen la misma coherencia interna. La aritmética produce los mismos resultados independientemente del idioma que se utilice para describir los componentes individuales de una expresión aritmética.

Los distintos sistemas numéricos son exactamente iguales que los diferentes idiomas. El número de los dedos de una mano no se modifica porque se diga *five* o *cinco*, ni tampoco se altera porque se escriba 5 o 101b (recuerde que la *b* significa número escrito en sistema binario). Las dos únicas razones por las que se escoge un sistema u otro son la costumbre o la conveniencia.

Si, de niños, nos convenía aprender la representación decimal era porque se trata del sistema numérico más comúnmente utilizado a nuestro alrededor. Pero no es el único sistema. Los relojes digitales, por ejemplo, emplean un sistema de aritmética caprichoso; parte decimal, parte módulo 60 (hay 60 minutos en una hora y 60 segundos en un minuto) y parte módulo 24 (24 horas en un día). Con anterioridad a 1971, la moneda británica se contaba por unidades de 12 (peniques por chelín) y 20 (chelines por libra). Aprender a utilizar estos sistemas costaba años de angustiantes ejercicios escolares.

Cuando hablamos de ordenadores, nos resulta instructivo empezar por hablar acerca de los números binarios porque ejemplifican muy estrechamente las operaciones eléctricas del ordenador, al ser simplemente secuencias de estados encendido-apagado. Si sólo deseáramos hablar de números de un solo byte, entonces el sistema binario nos podría servir de alternativa completa al decimal: traducir de binario de ocho bits a decimal se hace sorprendentemente sencillo después de un poco de práctica. Por desgracia, las direcciones de memoria en particular y los números utilizados en general son normalmente demasiado largos como para que quepan en un byte, de modo que con el transcurso de los años los programadores y los ingenieros de ordenadores han visto la necesidad de un sistema numérico que reúna la conveniencia lógica del binario y la amplitud del decimal. Esto sólo lo ofrecen dos sistemas: el hexadecimal y el octal. El primero, ahora estándar en microinformática, familiarmente conocido como *hexa*, tiene por base el número 16. El octal, de base 8, se usó en la informática de ordenadores de unidad principal, pero se está sustituyendo por el hexadecimal.

El sistema hexadecimal

Al analizar la representación decimal y la binaria, hemos visto que la elección de la base numérica tiene dos consecuencias: la base es el número de símbolos diferentes que necesita el sistema, y es el factor multiplicativo teniendo en cuenta la posición. Por ejemplo, en decimal hay diez dígitos exclusivos (0-9) y el valor de un dígito decimal se multiplica por diez cada vez que se desplaza hacia la izquierda en un número decimal (19 no es "uno-nueve", sino "diez y nueve").

El hexadecimal, por lo mismo, exige 16 dígitos exclusivos, que, por acuerdo, son los dígitos de 0 a 9 y las letras desde la A a la F. Contar en hexa es una simple cuestión de ir nombrando los dígitos únicos y una vez agotados volver a utilizarlos en notación posicional. El número hexa que sigue a 9, por tanto, es A (decimal 10); el siguiente es B; el siguiente C; y así sucesivamente hasta F (decimal 15). Esto agota los dígitos individuales, de modo que el número que sigue a F es 10 (que se lee: "uno-cero hexa"), que corresponde al decimal 16. A partir de esto podemos ver cómo se utilizan dos dígitos individuales y que el valor de los lugares en un número hexa de varios dígitos aumenta según un factor de 16 con desplazamiento hacia la izquierda. En un número decimal, a los lugares los llamaremos: unidades, decenas, centenas y millares. De modo semejante, en un número hexa los lugares son: unidades, "dieciseisenas", "doscientas-cincuenta-y-seisenas" y "cuatro-mil-noventa-y-seisenas". Comparando los cambios en la columna binaria con los cambios en la columna hexa, usted debería ser capaz de ver la principal ventaja de los números hexas: cualquier número binario de *cuatro* bits puede expresarse con un número hexa de *un único* dígito (o sea, de 0 a 15 en decimal). Esto quedará más claro con algunos ejemplos:



Decimal	Binario	Hexa
0	00000000	0
1	00000001	1
2	00000010	2
3	00000011	3
...	...	...
7	00000111	7
8	00001000	8
9	00001001	9
10	00001010	A
11	00001011	B
12	00001100	C
13	00001101	D
14	00001110	E
15	00001111	F
16	00010000	10
17	00010001	11
...	...	...
24	00011000	18
25	00011001	19
26	00011010	1A
27	00011011	1B
...	...	...
31	00011111	1F
32	00100000	20
33	00100001	21

Cualquier número binario de hasta ocho bits (un byte) puede quedar reducido a dos cifras hexas. Compárense las respectivas extensiones:

de 0	a 255	en decimal
de 00000000	a 11111111	en binario
de 0	a FF	en hexa

Por consiguiente, para convertir un número hexa a binario basta con transformar cada dígito hexa en un número binario de cuatro bits. Si un número de un solo byte se expresa como un número hexa de dos dígitos, entonces el dígito hexa más a la izquierda corresponde a los cuatro bits binarios situados más a la izquierda, mientras que el dígito hexa situado más a la derecha corresponde a los cuatro dígitos binarios situados más a la derecha. Si dividimos un byte de esta manera obtenemos dos *nybbles* (un nybble corresponde a medio byte). El nybble más a la izquierda, que se corresponde con el dígito hexa más a la izquierda, se denomina nybble superior o más significativo; y el nybble más a la derecha se denomina nybble inferior o menos significativo. He aquí un ejemplo:

```
                Nybble      Nybble
                Superior    Inferior
                   ↓           ↓
206       =       1100        1110   = C E
 ↑                      ↑                ↑
decimal   equivalente    binario     equiv. hexa
```

Es importante que nos familiaricemos con el sistema numérico hexadecimal, por la sencilla razón de que la manipulación de bytes de ocho bits resulta con él mucho más sencilla que utilizando el binario. Para que se convenza de ello bastará hacer un poco de práctica, no sólo con ejemplos numéricos, sino particularmente con direcciones y contenidos de bytes de memoria. Una vez comprendida su importancia (y se comprenderá muy pronto) usted se preguntará cómo le concedía tanta utilidad al decimal.

En este capítulo sobre lenguaje máquina le ofrecemos programas para el BBC Micro, el Commodore 64 y el Spectrum, que nos permiten adivinar el contenido de un byte elegido de la memoria. Estos programas "Mempeek", como los hemos llamado, primero le piden que enuncie la "dirección de comienzo" (es decir, que especifique el número del primer byte) y luego que dé el número de bytes a analizar. Si, por ejemplo, usted desea especificar el byte 1953 como su punto de partida y solicita que se visualicen los contenidos de los cuatro bytes siguientes, entonces la pantalla mostrará el número decimal 1953 en la columna de la izquierda, y luego listará los contenidos del byte 1953, byte 1954, byte 1955 y byte 1956 en las cuatro columnas siguientes.

Tenga presente que si la máquina muestra que el byte 1956 contiene el número decimal 175, lo que queremos decir es que en uno de los chips de memoria, una zona que la máquina denomina byte 1956 lleva un patrón de ocho niveles de voltaje. Si 0 voltios se representa con 0, y 5 voltios con 1, luego el byte 1956 lleva el patrón de voltaje 10101111. Éste es el que hemos escogido para interpretar como binario, y su equivalente decimal es 175.

Es de vital importancia recordar que la mayoría de las veces que hablamos de ordenadores utilizamos una especie de taquigrafía muy imprecisa, y ampliarla a una descripción física siempre es saludable y debería contribuir a evitar las confusiones.

El contenido de un byte visualizado en la pantalla no es el "real". Lo que vemos son datos de caracteres que se le han asignado a los patrones de voltaje de los bytes. Esto significa que habiendo interpretado los patrones de voltaje como números binarios, y habiendo convertido los números binarios en números decimales, estamos dando un paso más hacia adelante y convirtiendo números decimales en caracteres según el código ASCII (*American Standard Code for Information Interchange*: código norteamericano estándar para intercambio de información). Estos datos de caracteres se visualizan en la última columna de la visualización. Se trata de un código reconocido internacionalmente, que está incorporado en la mayoría de los ordenadores y que sustituye a los números decimales entre 0 y 127 para todos los caracteres de un teclado (históricamente, un teclado de teleimpresora). En este código, el número decimal 65 significa el carácter en mayúscula "A", el 66 significa "B", el 67 significa "C", y así sucesivamente. Entre los caracteres no alfabéticos, 32 significa un espacio, 42 significa un asterisco, 13 significa la tecla Return (retorno), y así sucesivamente.

Los caracteres ASCII imprimibles empiezan en el número 32 y terminan en el número 127. Los códigos fuera de esta escala no están definidos, o no son imprimibles, o son específicos de determinadas máquinas. Debido a ello, cuando ejecute los programas Mempeek, el monitor le imprimirá un punto para representar cualquier byte que contenga un número fuera de escala. En el próximo capítulo de este curso ofreceremos un amplio juego de caracteres ASCII para los valores comprendidos entre 0 y 127.

Una investigación del juego de caracteres ASCII es especialmente útil como trasfondo para una comprensión cabal del código de lenguaje máquina por dos importantes razones. En primer lugar, sirve para descubrir que la forma en que se interpreten los contenidos de la memoria es pura arbitrariedad. Se puede decir que un byte contiene un número, o

una dirección, o un carácter codificado, o una instrucción, o cualquier otra cosa que nos venga en gana. En cualquier caso, serán datos que esperan ser interpretados. En segundo lugar, proporciona una visión bastante más comprensible de la memoria, en especial de aquellas partes de la misma que realmente contienen datos de caracteres, algunos de ellos utilizados por el sistema operativo de la máquina y algunos otros empleados por el usuario.

Los datos del sistema operativo incluyen todos los mensajes de error y aviso: por ejemplo, READY, o NONSENSE IN BASIC (no significa nada en BASIC), o START TAPE THEN PRESS RETURN (accione cinta, luego pulse Return); todo cuanto sea capaz de decirle al usuario debe codificarse en ASCII y almacenarse en la memoria. Puede que no haya pensado nunca en esto, pero es una idea reveladora de las limitaciones de un ordenador como máquina "inteligente". Nuestra inteligencia es diferente: no memorizamos mensajes, sino que elaboramos un pensamiento y luego generamos una combinación adecuada de palabras para expresarlo.

Mapas de memoria

Un mapa de memoria es una representación esquemática de la distribución dada a la memoria junto con los parámetros de zonas específicas. Algunas zonas de la memoria siempre se utilizan con el mismo fin. En el Commodore 64, por ejemplo, desde el byte 0 hasta el byte 1024 los utiliza el sistema operativo de BASIC como área de trabajo. Otras zonas de la memoria poseen diversos usos según el tamaño y el estado del programa. Las fronteras entre estas zonas pueden ser *fijas* (en los diagramas que ofrecemos abajo aparecen como líneas continuas) o bien *flotantes* (las líneas a trazos). Las fronteras fijas no se alteran nunca, mientras que las flotantes son para aquellas áreas de la memoria que fluctúan a tenor de las necesidades. En el mapa de memoria del Commodore, las fronteras de la RAM de pantalla son fijas (en los bytes 1024 al 2048), y las de la zona de memoria donde se conservan las variables de BASIC fluctúan según la cantidad que se use en un momento dado.

Los programas Mempeek de la página siguiente se pueden utilizar para localizar las posiciones corrientes de las fronteras flotantes de la memoria de su máquina. El Commodore posee seis indicadores de fronteras flotantes (también denominados variables del sistema). En el cuadro inferior ponemos un ejemplo de cómo se emplean los contenidos de un par de bytes para calcular la dirección de memoria requerida. El BASIC del BBC posee cuatro variables del sistema para determinar y el Spectrum cinco.

Es necesario recordar que un mapa de memoria es una representación estática de algo que cambia de forma continuamente mientras se está utilizando la máquina. Cada una de las fronteras flotantes está sujeta a modificaciones en cualquier momento. Vea en la página siguiente algunas ideas para ampliar los programas Mempeek para observar las variaciones en los valores de los indicadores.

Las variables de sistema Commodore

- 43,44 Comienza el TEXTO DEL PROGRAMA EN BASIC
- 45,46 Comienzan las VARIABLES EN BASIC
- 47,48 Comienzan las MATRICES EN BASIC
- 49,50 Terminan las MATRICES EN BASIC
- 51,52 Parte inferior del ALMACENAMIENTO DE SERIES EN BASIC
- 55,56 Parte superior del ALMACENAMIENTO DE SERIES EN BASIC

Ejemplo

Utilice el programa Mempeek de p. 539 para inspeccionar los contenidos de estos bytes. Su visualización en pantalla podría tener el siguiente aspecto:

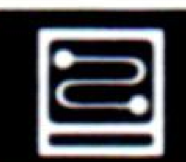
43 0 8 11 9

La primera columna es la dirección del primer byte al cual se ha accedido. La segunda y tercera columnas visualizan los contenidos del byte 43 y byte 44. Éstos son los bytes de desplazamiento y de página (véase p. 516) de la dirección de comienzo del área de texto en BASIC. Ésta se calcula así:

8*256 + 0 = 2048

La cuarta y quinta columnas de la visualización son los bytes de desplazamiento y página para el final del área de texto en BASIC. La dirección se calcula así:

9*256+11 = 2315

BBC MICRO

```
7 REM**********************************
8 REM*     MEMPEEK 1           BBC       *
9 REM**********************************
20 MODE 7
30 *TV 255
40 CLS
50 REPEAT
100 INPUT"DIRECCION DE COMIENZO ",DC
200 INPUT"NUMERO DE BYTES (0 PARA
    SALIR) ",NB
250 PRINT "**************************
    *********"
300 FOR B%=DC TO (DC+NB-1) STEP 4
350 H$="":@%=6
400 PRINT TAB(0);B%;TAB(8);
450 @% =4
500 FOR C=0 TO 3
550 PK%=? (B%+C):PK$="."
600 PRINT PK%;
650 IF PK%=13 THEN PK$=CHR$(124)
700 IF (PK%>31) AND (PK%<128) THEN
    PK$=CHR$(PK%)
750 H$=H$+PK$
800 NEXT C
850 PRINT TAB(32);H$
900 NEXT B%
950 UNTIL NB=0
1000 REM******************************
```

Spectrum

```
7 REM*****************************
8 REM*       MEMPEEK 1 SPECTRUM     *
9 REM*****************************
30 DIM H$(4)
50 FOR L=0 TO 1 STEP 0
100 INPUT"DIRECCION DE COMIENZO ";DC
200 INPUT"NUMERO DE BYTES (0 PARA
    SALIR) ";NB
250 PRINT "**********************
    *****"
300 FOR B=DC TO (DC+NB-1) STEP 4
350 LET H$="...."
400 PRINT B;TAB 7;
500 FOR C=0 TO 3
550 LET PK=PEEK(B+C)
600 PRINT PK;" ";
650 IF (PK> 31) AND (PK< 128) THEN
    LET H$(C+1)=CHR$ PK
700 IF PK=13 THEN LET H$(C+1)="■"
800 NEXT C
850 PRINT TAB 26;H$
900 NEXT B
950 IF NB=0 THEN LET L=2
1000 NEXT L
1050 REM************************
```

Use los Mempeek

Cuando dé entrada al programa Mempeek en su máquina, asegúrese de guardarlo (SAVE) y verifíquelo concienzudamente antes de ejecutarlo (RUN), porque en esta clase de programas los errores de digitación pueden producir roturas irreparables.

Primero el programa le solicitará una dirección de comienzo y después el número de bytes que usted desea examinar. Ambos deben ser números enteros positivos comprendidos entre 0 y 65535. Dar entrada a 0 como el número de bytes hará que el programa termine (salida). Supongamos que introduce como dirección de comienzo el byte 230. La visualización en pantalla podría asumir el siguiente aspecto:

```
DIRECCION COMIENZO? 230
NUMERO DE BYTES (0 PARA SALIR)? 8
**********************************

230  193  32  65  49   . A1
234  129  64  93  98   .@]b
DIRECCION DE COMIENZO?
```

La columna situada más a la izquierda proporciona la dirección decimal del primer byte, las cuatro columnas siguientes proporcionan los contenidos decimales de los cuatro bytes a partir de dicha dirección en adelante, y la última columna proporciona la representación en caracteres de los contenidos de los bytes, cuando esto sea posible; de lo contrario, dará '.'

Puede que opte por "vagabundear" con este programa a lo largo y ancho de la memoria, tomando nota de cualquier dirección interesante, y trate luego de hallar en qué lugar de la memoria almacena el sistema operativo sus mensajes de error y sus palabras clave en BASIC. Su manual para el usuario podrá serle de ayuda en esto.

Una vez hallados los indicadores que definen las fronteras de las diversas zonas de la memoria, puede tratar de agregarle al programa algunas líneas REM y ver el efecto que ello tiene en los valores de los indicadores. Después agregue algunas líneas al principio del programa para hacer alguna manipulación de series y, nuevamente, vea el efecto que produce en los indicadores y en los contenidos del área de almacenamiento de variables.

Por ejemplo:

```
3 DIM Z$(254)
4 LET X$ = ""
5 FOR M = 1 TO 255:LET X$ = X$ + "*":NEXT M
```

Commodore 64

```
7 REM*********************************
8 REM*      MEMPEEK 1 COMMODORE        *
9 REM*********************************
30 PRINT CHR$(147)      :REM LIMPIAR
9 FPANTALLA
40 PRINT CHR$(142)      :REM MAYUSCULAS
50 FOR LP=0 TO 1 STEP 0
100 INPUT"DIRECCION DE COMIENZO ";DC
200 INPUT"NUMERO DE BYTES (0 PARA
    SALIR) ";NB
250 PRINT "************************
    *************"
300 FOR B=DC TO (DC+NB-1) STEP 4
350 H$=""
400 PRINT B;TAB(8);
500 FOR C=0 TO 3
550 PK=PEEK(B+C):PK$="."
600 PRINT TAB(8+5*C);PK;
650 IF PK=0 THEN PK$=CHR$(122)
700 IF (PK> 31) AND (PK< 128) THEN
    PK$=CHR$(PK)
750 H$=H$+PK$
800 NEXT C
850 PRINT TAB(32);H$
900 NEXT B
950 IF NB=0 THEN LP=1
1000 NEXT LP
1050 REM**************************
```

De la bellota a la encina

Acorn Computers produce dos excelentes ordenadores personales: el BBC Micro y el Electron


Chris Curry


Herman Hauser

Cortesía de Acorn

El fundador de Acorn, Chris Curry, era un empleado y amigo de sir Clive Sinclair. Curry se había incorporado a la Sinclair Radionics en 1965, cuando Sinclair le ofreció un puesto como ingeniero de proyectos nuevos con un estipendio semanal de 11 libras.

En la Sinclair Radionics, Curry se hizo cargo del proyecto de investigación que en 1971 produjo la calculadora Executive. Durante los cinco años siguientes, se dedicó a desarrollar calculadoras, de factura tal que hoy son consideradas como las precursoras del moderno ordenador personal. En 1975, la Sinclair Radionics dejó de producir y Curry se unió a Sinclair en una operación independiente llamada Science of Cambridge. Esta nueva empresa se proponía reunir los componentes electrónicos en conjuntos de piezas para montar y venderlos como *kits*.

Una idea de mucha aceptación fue el reloj de pulsera con calculadora. Pero Curry también se sintió atraído por los ordenadores de un solo tablero que comenzaban a surgir en Estados Unidos, y se propuso crear su propio modelo desmontable. Se llamó MK14 (microprocesador en forma de kit de 14 chips) y constaba de un microprocesador National Semiconductor, con 256 bytes de RAM, de una pequeña memoria fija que contenía el monitor, y de los componentes necesarios para la alimentación eléctrica de una pantalla LED (*Light Emitting Diode*: diodo emisor de luz) de ocho dígitos.



Cortesía de Acorn

Curry comprobó que la empresa constantemente estaba proporcionando consejos e ideas por teléfono a aficionados a la electrónica, y decidió proponer a Herman Hauser, que estudiaba el doctorado en filosofía en la Universidad de Cambridge, para que atendiera estas consultas. Pronto, sin embargo, los planteamientos de Curry empezaron a disentir de los de Sinclair, y pensó que bien podía ser hora de crear una empresa propia. Con Hauser como nuevo socio, Curry formó una empresa llamada Cambridge Processor Unit (un nombre escogido con picardía, ¡ya que sus siglas eran CPU!). Desde un pequeño despacho situado en Bridge Street (Cambridge), ambos ofrecían sus servicios como consultores en electrónica y ordenadores.

El éxito del MK14 y las novedades llegadas de Estados Unidos demostraban claramente que lo que deseaban los clientes era un ordenador dentro de una caja con BASIC en el tablero. Ya que habían escrito una versión rápida de BASIC para control de la máquina en uno de sus trabajos de consulta, CPU decidió incorporarla a una máquina y sacarla al mercado. La máquina recibió el nombre de Atom (átomo) y CPU adoptó Acorn (bellota) como nombre comercial para la empresa encargada de su lanzamiento al mercado. Se pretendía que la máquina captara sobre todo el mercado educativo, pero la mayoría de las escuelas creían que su BASIC se apartaba demasiado de la versión Microsoft como para que fuera aceptable. No obstante, la máquina fue recibida con mucho entusiasmo por parte de los aficionados. Acorn siguió adelante con una nueva versión del Atom, que recibió el nombre de Proton (protón), destinado a su utilización en laboratorios y colegios.

Pero en 1981, con el Proton en fase de prefabricado, Curry se enteró de que la BBC buscaba una máquina para su programa de alfabetización informática. Curry demostró a la BBC las capacidades del procesador 6502, pero no en el Proton sino en un sistema diseñado especialmente.

Las indicaciones de la BBC aludían a una máquina fácil de manejar por los principiantes pero que se pudiera ampliar para conformar un estándar muy elevado. La máquina debía ofrecer además una buena relación calidad-precio (la compañía de radiotelevisión británica inicialmente hablaba de un precio orientativo de 200 libras). Con la oposición de Sinclair Research, el trabajo se encomendó a Acorn, que creó el BBC Micro, del que hoy se produce al mes un promedio de 12 000 unidades.

UNION PERFECTA

Así se comportan los periféricos creados por SINCLAIR para SINCLAIR: de forma perfecta. Y es lógico.

Cada vez que SINCLAIR diseña un microordenador, no lo hace de una manera aislada. Simultáneamente crea todos esos periféricos que van a hacer más potente, preciso y útil el microordenador que tiene entre manos.

Periféricos pensados y diseñados para dar un servicio óptimo, pero con un precio razonable, dentro de la filosofía SINCLAIR:

"Hacer la informática accesible a todos".

Por eso cuando creó el ZX 81 vio la necesidad de dotarlo con una ampliación de memoria de 16K RAM para que no quedara pequeño y de una impresora sencilla y barata pero útil y precisa.



Así es la filosofía SINCLAIR. Así son los periféricos de SINCLAIR para SINCLAIR.







9 788485 822836
10027